40261

National Library of Canada / Bibliothèque nationale du Canada

CANADIAN THESES ON MICROFICHE / THÈSES CANADIENNES SUR MICROFICHE

NAME OF AUTHOR/*NOM DE L'AUTEUR* Martin Nohr

TITLE OF THESIS/*TITRE DE LA THÈSE* A Microprocessor Controller System

UNIVERSITY/*UNIVERSITÉ* University of Alberta

DEGREE FOR WHICH THESIS WAS PRESENTED/ *GRADE POUR LEQUEL CETTE THÈSE FUT PRÉSENTÉE* M. Sc.

YEAR THIS DEGREE CONFERRED/*ANNÉE D'OBTENTION DE CE GRADE* 1978

NAME OF SUPERVISOR/*NOM DU DIRECTEUR DE THÈSE* Dr. Y.J. Kingma

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

DATED/*DATÉ* Oct. 20, 1978 SIGNED/*SIGNÉ* Martin Nohr

PERMANENT ADDRESS/*RÉSIDENCE FIXE* 370 - 11 St. SE

Medicine Hat, Alberta

NL-91 (3-74)

National Library of Canada

Cataloguing Branch
Canadian Theses Division

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada

Direction du catalogage
Division des thèses canadiennes

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

THE UNIVERSITY OF ALBERTA

A MICROPROCESSOR CONTROLLER SYSTEM

by

© Martin Nohr

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Department of Electrical Engineering

EDMONTON, ALBERTA

Fall, 1978

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled A MICROPROCESSOR CONTROLLER SYSTEM submitted by Martin Nohr in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE in ELECTRICAL ENGINEERING.

.....

Supervisor

.....

.....

Date Oct 4th 1978

ABSTRACT

The continuing decrease in the cost of microprocessors as well as an increase in their capabilities has resulted in dedicated, on-line digital controllers becoming more attractive.

A microprocessor based on-line digital controller is described. For analog control it features the standard PID and D(z) controllers, along with two other novel control algorithms. For digital stepper motor control there is a direction output and a pulsed output. Both position and velocity algorithms are implemented for the stepper motor control. The controller is designed for ease of use so that sampled-data controllers may be conveniently studied without extensive knowledge of computer programming.

The instrument features sampling periods from 10 msec to 30 seconds. The coefficient magnitude range is from 0.000001 to 999. The control algorithms and coefficients are easily changed by keyboard entry. The controller can be used with either an internal or external set-point.

Simple digital filters can also be implemented on the device.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the assistance and encouragement of Professor W. J. Kingma in the preparation of this thesis. The author also wishes to acknowledge the assistance of fellow graduate students L. J. Davis and K. M. Weiss, along with several professors in the Department of Electrical Engineering.

The author gratefully acknowledges the financial assistance rendered by the National Research Council and the Department of Electrical Engineering.

Many thanks are also due the author's wife for her understanding and perseverance.

Table of Contents

Chapter Page

1. Introduction....1
2. System Implementation....6
   2.1 The Digital Controller....6
   2.2 The Hardware....8
      2.2.1 The Microcomputer....9
      2.2.2 System Memory....9
      2.2.3 Analog and Digital Conversion....10
      2.2.4 Timing Generator....11
      2.2.5 Keyboard and Display....12
   2.3 The Software....13
      2.3.1 Restart Program....13
      2.3.2 Foreground Program....14
         2.3.2.1 Fast Multiply Program....15
         2.3.2.2 Decimal to Hex Conversion....17
         2.3.2.3 Other Subroutines....17
      2.3.3 Controller Routines....18
3. Control Mode Implementation....19
   3.1 PID Implementation....19
   3.2 D(z) Implementation....21
   3.3 Dual-Mode Controllers....25
      3.3.1 Dual Mode with D(z)....26
      3.3.2 Dual Mode with Table Look-up....26
   3.4 Stepper Motor Control Implementation....27
      3.4.1 Positional Stepper Motor Control....28

3.4.2 Velocity Stepper Motor Control.....29

4. Controller Design.....31

4.1 PID design.....31

4.2 Dead-Beat Design.....32

4.2.1 Dead-Beat Controller Pulse Transfer Function Derivation.....33

4.2.2 Computer Program to Calculate Dead-Beat.....35

4.3 Dual Mode Design.....36

4.4 Digital Filter Design.....37

5. Controller Tests.....40

5.1 PID Mode Tests.....41

5.2 D(z) Mode Tests.....42

5.3 Dual-Mode Tests.....43

5.3.1 Test with Plant Pole Variation.....43

5.3.2 Test with Noise Added to Loop.....43

5.4 Stepper Motor Control Tests.....44

5.4.1 Positional Stepper Motor Test.....44

5.4.2 Velocity Algorithm Stepper Motor Test.....44

Conclusions.....59

Future Plans.....60

Bibliography.....61

Appendix.....62

Operators Manual.....63

Short Form Operators Manual.....73

System Memory Map.....74

System Flowcharts.....75

Schematic Diagrams.....85

The Program Source Code.....90

List of Figures

Figure		Page
1.	Digital Control System	6
2.	Internal Set Point Controller	7
3.	Controller Hardware	8
4.	Floating Point Storage Format	15
5.	Digital PID Control Loop	20
6.	Direct Canonic Form	22
7.	Error vs Frequency for Stepper Control	29
8.	State Variable Transfer Function	34
9.	Plant Transfer Functions	41
10.	Integrator Ki=0.1 T=1 S	46
11.	Integrator Ki=0.001 T=10 mS	46
12.	Differentiator Kd=2 T=0.5 S	47
13.	Differentiator Kd=2 T=50 mS	47
14.	P-I Control of System A	48
15.	System A Kp=1 T=0.1 S	48
16.	System A Kp=1 T=1 S	49
17.	System A Kp=1 T=2 S	49
18.	System A Kp=1 T=4 S	50
19.	Dead-Beat Control of System A	50
20.	Dead-Beat Control of System D	51
21.	Dead-Beat Control of System C	51
22.	Dual-Mode Control of System A	52
23.	Dual-Mode Control of System B	52
24.	D(z) Dual-Mode with pole at -1.25	53
25.	Table Dual-Mode with pole at -1.25	53

26. D(z) Dual-Mode with pole at -0.75.....54
27. Table Dual-Mode with pole at -0.75.....54
28. Table Dual-Mode Control of B with Noise.....55
29. D(z) Dual-Mode Control of B with Noise.....55
30. Stepper Motor Positional Response.....56
31. Velocity Algorithm, Proportional Output.....56
32. Velocity Algorithm, Integral Output.....57
33. Velocity Algorithm, Derivative Output.....57
34. Velocity Algorithm, PI Control.....58
35. Controller Keyboard.....63

1. Introduction

Soon after the development of the digital computer it was recognized as a powerful instrument for implementing control systems. The digital computer can not only implement the control algorithm, but it can also easily check the system for overloads, keep a log of the plant which is under control, and even optimize the control algorithm. More complicated control algorithms can be implemented on a digital computer than is practical with hardwired controllers.

Another important advantage of using a computer as a controller is the advantage of digital control. The parameters of a digital controller are stored as words in memory locations and are therefore not subject to variations due to age or temperature. The digital controller is adjusted for the specific plant under control and need only be readjusted if the plant changes.

Due to these and many other advantages computers began to be used for on-line control in many industries. A major disadvantage was the high initial cost of a large computer. In order to spread the cost the computer had to be used for other business functions as well as plant control. Book keeping programs would use the computers process time and make less time available for running time-critical control algorithms.

Another disadvantage was using only one computer in a plant to run all of the control loops. If then the computer operation failed, all loops were without control.

A far better solution would be to have a dedicated computer for each control loop. The introduction of mini-computers was a step in this direction. Many minis could be purchased for the cost of one large installation. For many processes, however, the advantages could still not justify the cost.

The development of micro-computers and the sharp decreases in their cost has now made dedicated digital controllers attractive and economically practical.

As digital controllers become more common, it is important that electrical engineers have a basic understanding of digital sampled controllers. Undergraduates are capable of understanding the theory of digital control[1] and need only the experience of designing and measuring some digital control algorithms. The drawback in the past has been the equirement of a large programming effort on the part of the student. What is needed is a calculator like device that can perform digital control algorithms, with the various coefficients entered from a keyboard. This approach allows the engineer to treat the controller as a black box with a transfer function that he can select by keyboard entry.

---

1 Martens, "IEEE Transactions on Education", Vol. E-11, No. 4, Sept. 1968

A device of this black box nature is also useful for experimenting with different digital controllers in the laboratory. The dead-beat, or ripple free minimal time response, controller can be easily implemented and the effect of control or plant parameter changes observed. Other novel control algorithms, such as automatic switching between dead-beat and PID, can be conveniently studied in the laboratory.

A digital controller which has analog inputs and outputs can also be used as a digital filter. Digital filters are very useful for low frequency use, as there are no capacitors or inductors which would become prohibitively large for very low frequency use. Since the digital filter tuning is done by constants contained in memory a digital filter is easily duplicated, with no manual tuning needed.

Signal conditioning of very low frequency signals is often needed in biological research. The digital filter is very useful in this application.

This thesis describes the design of a useful calculator like digital controller. The device was developed to meet the following objectives.

1) Standard ±10 volt analog ranges, with a resolution of 12 bits (4.88 mV)
2) Single input and output
3) No extensive user programming ability required
4) Coefficients easily entered from a keyboard, and displayed on a digital display

5) Controller algorithm selected from the keyboard
6) Sampling rate selected from the keyboard
7) Sampling rate as fast as possible to widen the useful digital filter range
8) Useable with internal or external set point
9) Derivative control turned off for set-point change in the PID control mode (bumpless transfer)
10) Use a minimum of memory (less than 4k of program storage, and 1k of random access storage)
11) Position and velocity algorithm for stepper motor control

Described is a micro-processor based black box controller which is useful for laboratory digital control experiments. Several different controller types were programmed into the digital controller. The following control algorithms were implemented and tested.

1) Standard PID controller
2) D(z) type controller
3) Positional stepper motor control
4) Velocity stepper motor control
5) A dead-beat controller which uses a look-up table to output the control whenever the set-point is changed, and otherwise runs in the PID mode
6) Same as the above but using D(z) for the dead-beat response

The above two last control algorithms were investigated with noise added into the control loop. They were also tested in their response to changes in the controlled plant parameters.

2. System Implementation

2.1 The Digital Controller

As seen in figure 1, a digital sampled control system consists of a sampler, an analog to digital converter (A/D), the micro-computer, a digital to analog converter (D/A), and a first order hold circuit. The plant transfer function G(s) can be implemented on an analog computer.

The configuration of the system actually developed differs slightly from this. The sampler does not exist externally as a separate circuit element, but is contained implicitly in the computer. The sampling operation is performed by the computer reading the A/D periodically. Another difference is the hold circuit which does not exist as a separate circuit element following the D/A. Instead the D/A converter is driven from a latch, which results in the

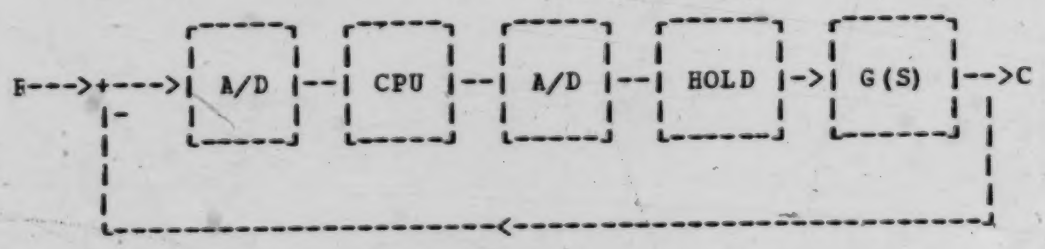


Fig. 1 Digital Control System

same effect as the hold circuit after the D/A. The final difference is the error signal subtraction. This operation is done internally in software after the A/D instead of before it. Thus the value read by the A/D is C and not R - C as in figure 1.

As seen in figure 2, the controller is normally used with its internal set-point. The controller can be used with an external error subtractor by simply leaving the internal set-point set to zero. Note that this introduces a sign change into the control algorithms since the A/D sample is subtracted from the set-point. The practical digital controller requires various blocks in order to be functional and convenient to use. The main hardware blocks, seen in figure 3, are the keyboard, the micro-processor, the A/D converter, the D/A converter, the timing generator, and the digital display. As well, the processor requires memory, and

Fig. 2 Internal Set Point Controller

power. There are two types of memory in the system, read/write random access memory (RAM), and read-only program memory (ROM).

2.2 The Hardware

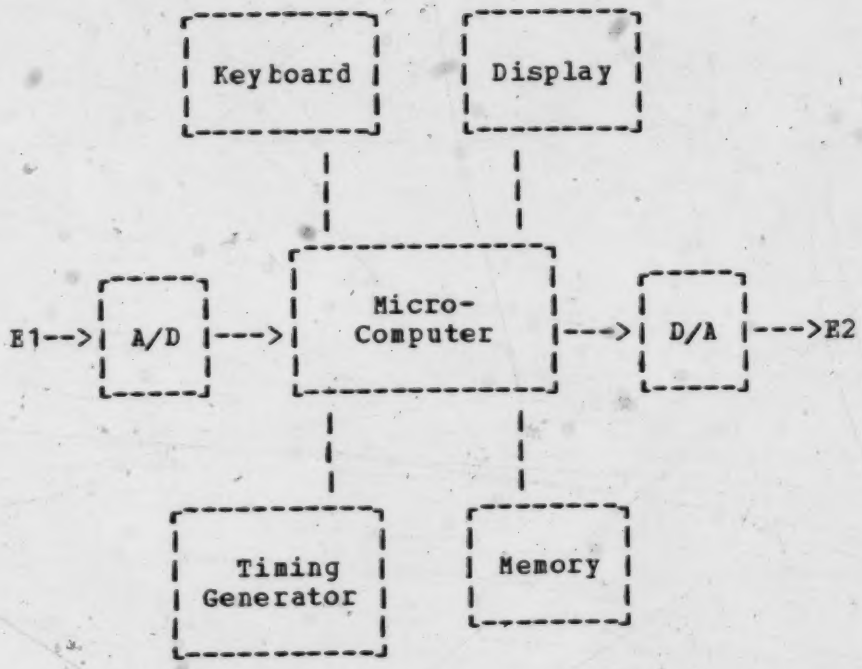


Fig. 3 Controller Hardware

2.2.1 The Microcomputer

It was decided to use a commercial micro-computer board as the central block of the system. This board was chosen as it contained the necessary circuit elements for the processor part of the controller. The A/D and D/A converters and the timer circuit were added on another board. The processor board chosen is a product of Motorola and has the following main features.

1) MC6800 micro-processor integrated circuit (IC)
2) 48 Input-Output lines (IO), and 12 IO control lines to be used for the D/A and A/D converters
3) 1024 words of RAM
4) Sockets for 4096 words of ROM

2.2.2 System Memory

The ROM's used for the program memory are 1024 by 8 bits in size, and are eraseable by exposure to short wave UV radiation. This is a convenient memory to use for program development. It is non-volatile so that the memory remains after the power is removed, but it can easily be erased for program changes during the operating system design. The complete operating system for the digital controller fits in less than 3k of ROM.

The 1024 words of RAM included on the micro-computer board are more than sufficient for the operating system requirements. The highest addresses are used by the system

for a push-pull type stack, also called a last-in-first-out buffer (LIFO). This structure is useful for saving the return address of a subroutine call as well as intermediate results of calculations. The stack structure is also used to save the processor register values and return address on an interrupt. This allows reentrant routines to be used. For an example see the multiply routine described in the software section.

The lowest addresses are used as fixed address data storage locations by the various subroutines in the system.

2.2.3 Analog and Digital Conversion

Since the desired resolution of the system was 12 bits, compatible 12 bit A/D and D/A units from Burr-Brown were chosen. This number of bits results in a very low level of quantization noise. These converters feature bipolar outputs to give the ±10 volts and are complete, requiring a minimum of external components for operation.

These converters can be wired to use twos complement binary numbers. This number system is convenient for the mathematical routines used in the controller system.

The micro-computer board contains 6 - 8 bit IO ports along with 24 IO control lines. Two of these 8 bit ports are used as the input to the D/A converter. These port lines are latched and therefore serve as the first-order hold circuit. Another two 8 bit ports are used to read the result of an

analog to digital conversion from the A/D. One control line is used to start the A/D conversion, another control line is used to signal the completion of the conversion.

Note that in order to maintain the accuracy of the 12 bits analog and digital conversions it is necessary for the 8 bit CPU to perform calculations in double precision.

2.2.4 Timing Generator

A Motorola MC6800 compatible timer IC (MC6840) was chosen to generate the sampling time intervals. This IC contains 3 separate 16 bit counters which are easily controlled from the system software.

One of the 16 bit counters is preset and used to divide the 1 MHz system clock to 1 ms pulses. Another one of the counters is then used to count the 1 ms pulses to generate the desired sampling interval. This results in a sampling time resolution of 1 ms. The output of this counter is also used to generate sampling interrupts to the processor. 16 bits could be used to count to 65.536 Seconds, but it was decided to limit the maximum time to 30 Seconds. This is not felt to be a limitation as there are few applications for sampling times longer than 30 seconds.

There is also a lower limit of 10 ms on the sampling time. This limit is due to the time required by the system to perform the control calculations.

The third timer is used for the stepper motor control.

Depending on the control mode this timer is used either as a variable frequency generator or to output a predetermined number of pulses at a given rate.

2.2.5 Keyboard and Display

To simplify the interface to the keyboard and the digital display unit, it was decided to use the INTEL 8279 display and keyboard interface IC. This IC contains most of the electronics for reading a keyboard and controlling a digital display.

The keyboard is a 4 by 6 matrix, divided into function keys, number keys, and a master reset button. Their operation is fully described in the appendix.

All coefficient values are entered into the system in normalized scientific notation, with a 3 digit mantissa and a one digit exponent. This requires 6 digits for the display, which is also sufficient for error messages.

Seven-segment displays were chosen to display coefficients, sampling times, control modes, and other information necessary for the user of the digital control system. The appendix contains a complete users guide to the controller. The guide gives the necessary information for loading coefficients, selecting control modes, etc. It also defines the various error messages that the system displays due to attempted illegal operations.

2.3 The Software

Since the digital controller must output the new control value as quickly and as close to the correct time as possible , the actual control program is interrupt driven. The timer chip generates interrupts to the computer periodically, and the control program is called to take the input sample and calculate the control output. This interrupt has the highest priority, and takes precedence over any other operation. A second source of interrupts is from the stepper motor control. When the velocity algorithm has calculated a number of steps for the motor a counter is loaded. This counter causes interrupts at the stepper pulse rate and is decremented at each interrupt until it is equal to zero. When the micro-processor is free from a control interrupt it checks the keyboard to see if any keys have been pressed. If any button have been pressed, the proper action is taken.

The operation of the function and number keys is described in the appendix. The appendix also contains flow charts for the controller system and the source code for the program.

2.3.1 Restart Program

On either power-up or a master reset the initialization routine is called. This routine performs the following.

1) Initialize the RAM
2) Clear the display

3) Turn off the sample timer
4) Set stepper pulse rate to 10 ms
5) Initialize the D/A port as outputs
6) Initialize the A/D port as inputs
7) Output 0 on the D/A
8) Select PID mode with external set-point
9) Wait for commands from the keyboard

2.3.2 Foreground Program

Following system initialization the program goes into a loop waiting for buttons to be pressed. When a button is pressed the 8279 IC debounces it and puts its row and column into a first-in-first-out buffer (FIFO). The program checks this buffer and when a button is found it reads it and compares its location with a table entry. If the button is a number button and no function has previously been called that is expecting a number as an operand, it is translated into seven-segment form and written out to the digital display. If it is a function button, it is used to index a table of system subroutines and the corresponding one is called. If another function button is pressed while a previously called function is waiting for a number, an error message is put onto the display. All function formats and error messages are described in the user's manual, found in this thesis.

2.3.2.1 Fast Multiply Program

It was decided to perform the multiplications needed by the controller programs in the system software. There is hardware available to perform the floating point operations required , but the cost is high. Floating point calculations are necessary in order to make the controller a useful laboratory device.

This choice creates a constraint on the system since available floating point software requires 10 ms to perform a multiplication. Since the input value, and the output value are 12 bit numbers, only the coefficients need be in floating point form. This introduces some freedom into the multiply program and it does not need to produce a normalized floating point result, only a 12 bit binary number. With this freedom, the multiply routine can be speeded up.

The format used for storage of floating point numbers is shown in figure 4. Bit 7 of byte 0 is the mantissa sign

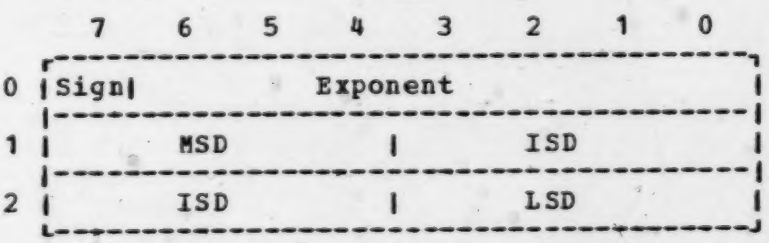


Fig. 4 Floating Point Storage Format

bit. Bits 6-0 are the exponent in 2's complement form. The next two bytes are the mantissa, consisting of 4 hexidecimal digits, in unsigned form. The mantissa can range from 0.0000 to 0.FFFF, and the exponent can range from -64 to +63. In decimal the coefficient magnitude can thus range from 8.64 EXP(-82) to 7.24 EXP(75). This range is far larger than is required for practical use due to the limited resolution of the D/A and A/D. It will therefore be reduced to allow for easy conversion from decimal to hexidecimal form, and also to speed up the multiply routine.

The D(z) transfer function implemented is particularly sensitive to arithmetic noise. This noise comes from quantization of the analog input and output, roundoff of results, and quantization of coefficients. The arithmetic routines will handle 16 bits, 4 more than the D/A and A/D. This "temporary overflow property" minimizes the danger of overflows occuring in intermediate calculations. This is especially important for D(z) calculations where the internal magnification could cause overflows.

The multiply program is written to take a 16 bit 2's complement binary number, multiply it by a floating point hex number, and produce a 16 bit 2's complement binary number. If an addition or multiplication were to overflow there is a possibility that a limit cycle could occur in the D(z) function. This is avoided by saturation arithmetic. On any overflow the result is set to the maximum value.

The system reduces the final output number to 12 bits,

as the D/A and A/D will not accept any more bits. Overflows will cause the output voltage to saturate at ±10. The system program insures this by detecting overflows after arithmetic operations and setting the result to its maximum value. The flowchart for the multiply routine is in the appendix section on flowcharts.

2.3.2.2 Decimal to Hex Conversion

As controller coefficients will normally be calculated in decimal by the user, the system must accept floating point decimal and translate it to the internal floating point hex notation. Numbers are entered into the controller display in normalized scientific notation. The fractional mantissa contains three digits plus a sign, and the exponent is one digit plus the sign.

i.e. ± .nnn ± (exp) Where nnn is the fractional mantissa and (exp) is the exponent digit. Note that the + sign is not displayed.

The decimal to hex program will accept mantissas from .001 to .999 and exponents from -3 to +3. This number range is sufficient for practical use of the digital controller.

2.3.2.3 Other Subroutines

There are several other subroutines required by the controller.

1) 7 segment to binary translation, used for loading coefficients read from the display
2) Binary to 7 segment translation, used for

displaying numbers

3) 16 bit 2's complement add program

2.3.3 Controller Routines

All controller programs are interrupt driven and form the background program. An interrupt to the processor is generated at each sampling instant. The system first starts the A/D conversion. After the A/D conversion is completed it is subtracted from the set-point and stored in memory for the control algorithm. The program then jumps to the subroutine for the selected control mode. (See the flowchart in the appendix) The development of the control subroutines is detailed in the next chapter.

3. Control Mode Implementation

The digital controller has its own internal setpoint, or as mentioned before, an external setpoint and subtractor can be used. The system subtracts the A/D result (E1) from the setpoint in memory and then performs the selected control mode calculations. The result of the calculations is output to the D/A.

This digital controller is designed to be used in the series control mode. The controller is cascaded with the plant, and the output is fed back to the controller input. This configuration is seen in figure 2. If desired, however, the controller can also be used in the parallel form as a feedback controller. In this case an external error subtractor must be used. Note the sign change through the controller when the internal set-point is not used.

3.1 PID Implementation

In analog plant control applications the PID (proportional-integral-derivative) controller is well known. While it is normally implemented using analog techniques, it can easily be programmed into a computer controller as a discrete PID algorithm. The digital PID controller is shown in figure 5.

The output of the PID is given by,

$$E2(k) = Kp\ e_{21}(k) + Ki\ e_{22}(k) + Kd\ e_{23}(k)$$

Where the components are,

Proportional part, $e_{21}(k) = e_1(k)$

Integral part, $e_{22}(k) = e_{22}(k-1) + T\ e_1(k)$

Derivative part, $e_{23}(k) = T^{-1}\ [e_1(k) - e_1(k-1)]$

At each sampling instant these calculations are performed and the control is output. The maximum time delay from the sample to the output is 3 ms.

In this form the derivative term is extremely sensitive to input noise. To reduce this an averaging technique is applied to the derivative values by taking the present and

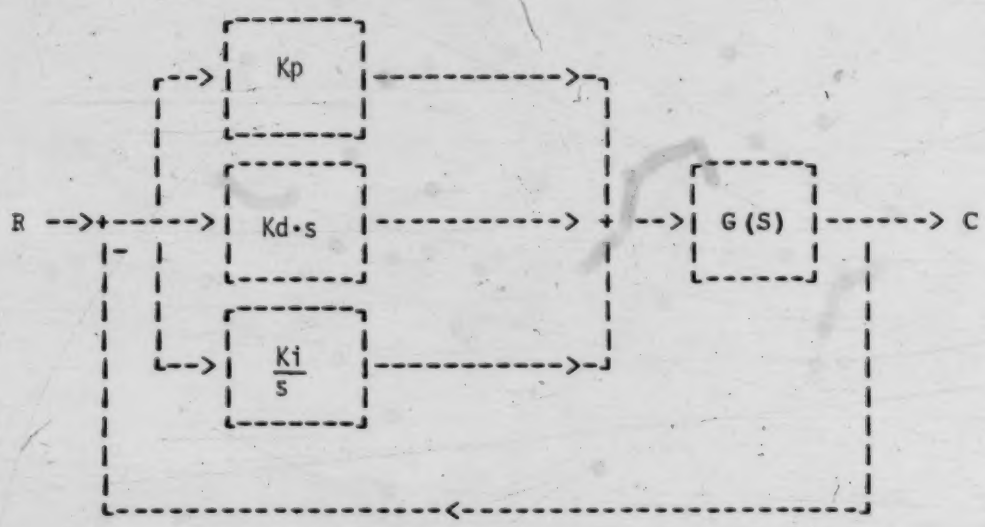


Fig. 5 Digital PID Control Loop

the previous three derivative terms, adding them and dividing by four to produce the actual derivative term which is used in the output summation.

The integral term is also sensitive to noise and quantization errors, but due to the closed loop nature of the system these errors will tend to zero in the limit.

In the chapter on controller tests photographs are shown illustrating the PID elements for different sampling times and gains.

3.2 D(z) Implementation

The D(z) control algorithm is natural to the computer due to its discrete nature. D(z) transfer functions can be found for PID elements by taking the z-transform. An interesting use of the D(z) controller is to implement the dead-beat or ripple free minimal time response controller. The design of such a controller is discussed in the controller design chapter, section 4.2.

The D(z) transfer function takes the following form.

$$D(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}}{1 + b_1 z^{-1} + \ldots + b_n z^{-n}}$$

The discrete transfer function can always be put into this normalized form. The condition for realizability is that

$a_n \neq 0$ unless $b_n \neq 0$

There are five forms in which the D(z) transfer function can be realized. Each form has disadvantages and advantages, which must be weighed for each use. It is thus difficult to say that any one form is always the best. The forms are;

1) Direct non-recursive
2) Direct recursive
3) Direct canonic
4) Parallel canonic
5) Cascade canonic

A realization is said to be non-recursive if no previous outputs are used in the calculation of the new output at a sampling instant. This is the case if all of the

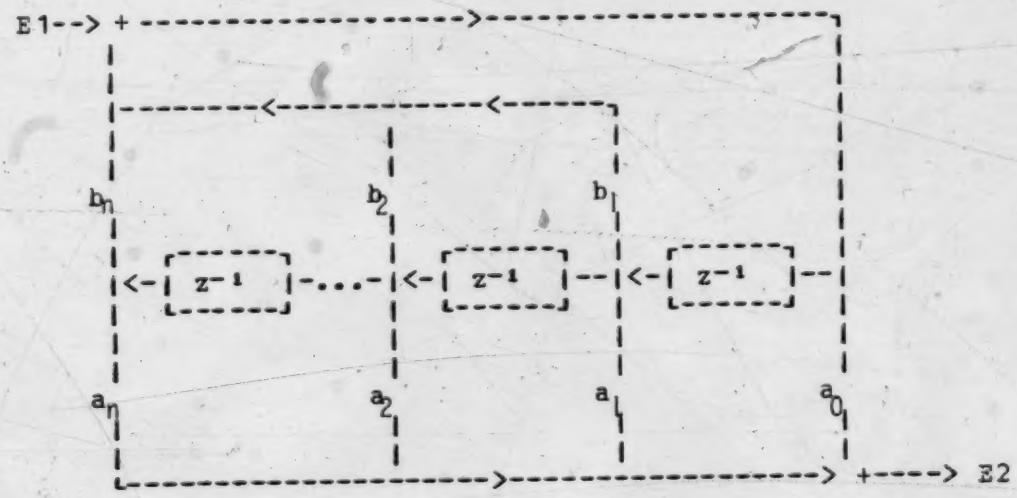


Fig. 6 Direct Canonic Form

b's in the transfer function are zero. If any one of the b's are non-zero then the function is recursive, and previous output values are used to find the new output.

The first two realization forms require more storage and more calculations than the canonic forms. The direct canonic form has the advantage of providing the new control output after one subtraction, one multiplication, and one addition. The remaining calculations can be done after the present output, but before the next output is required. This is a definite advantage in our system since the calculations are done in software and the output to the D/A has to occur as soon as possible after the sampling time interrupt.

The direct canonic form has the disadvantage of being sensitive to arithmetic roundoffs and quantizations. In particular, the poles of the function can move such as to make a stable system unstable.

e.g. If a 3 bit binary system tried to represent a pole at 15/16 it would be rounded off to 1. This occurs since 3 bits can only represent 7/8 or 1. In the z-domain a pole at z=1 is an integrator and therefore the system might become unstable.

This effect is especially bad for large order systems and this is the reason that the direct canonic form is not used in general for digital filters. It is not deemed to be a serious problem in our system since the system order is limited to 3, and also since the arithmetic routines have four extra bits compared to the D/A and A/D converters.

four extra bits compared to the D/A and A/D converters. Information and examples of noise problems in digital filter systems can be found in an article in EDN[2].

To program the direct canonic recursive form first multiply the numerator and denominator of the transfer function by a polynomial X(z). Then separating the numerator and denominator we have,

$$E2(z) = (a_0 + a_1 z^{-1} + \ldots + a_n z^{-n})\ X(z)$$

and

$$E1(z) = (1 + b_1 z^{-1} + \ldots + b_n z^{-n})\ X(z)$$

Rewriting the denominator,

$$X(z) = E1(z) - b_1 z^{-1}\ X(z) - \ldots - b_n z^{-n} X(z)$$

The polynomial X(z) can be written as;

$$x_1(z) = z^{-1}\ X(z)$$

$$x_2(z) = z^{-2}\ X(z)$$

$$\vdots$$

$$x_n(z) = z^{-n} X(z)$$

Since the operator z^{-1} represents a time delay, all calculations involving z^{-1} can be done after the control output. Defining,

$$G = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$$

and

$$H = - b_1 x_1 - b_2 x_2 - \ldots - b_n x_n$$

The calculation performed at the sampling instant then

---

[2] B. J. Leon and S. C. Bass, "Designers Guide to: Digital Filters", EDN, June 20, 1974

becomes,

$$X = E1 - H$$

and the output is calculated from;

$$E2 = a_0 X + G$$

G and H are calculated after the control is output to the D/A.

The signal flow graph for the direct canonic form is shown in figure 6.

3.3 Dual-Mode Controllers

An interesting controller type to investigate is one which attempts to combine the advantages of the PID control with the fast time response of dead-beat control for quick set-point changes. This controller would normally operate in the PID mode to give good plant control and noise rejection. PID control is also less sensitive to plant parameter variations than the dead-beat type.

This control mode can only be used with the internal set-point since it must be known when a set-point change occurs. The controller switches over to the dead-beat routine when a set-point change is entered from the keyboard. The sampling time is also changed since the dead-beat controller would normally require a longer sampling period than the PID controller in order to restrict the magnitude of the control signals.

For a step input, the dead-beat response can be completed in N sampling periods, where N is the order of the

intervals the controller switches back to the PID control and its original sampling time.

If the plant does not contain a free integrator, it is necessary that the controller contains one in order to eliminate steady-state error. A PI controller will therefore have zero steady-state error. In order for the dual-mode controller to properly switch back to the PID mode after a dead-beat sequence, it will be necessary to load the last output into the controller integrator.

3.3.1 Dual Mode with D(z)

When operating in this mode the dead-beat response is implemented with the D(z) transfer function. The coefficients are calculated according to the derivations in the part on controller design or by a computer program on a larger machine.

3.3.2 Dual Mode with Table Look-up

The dead-beat response can also be accomplished by simply outputting N control steps, one at each sampling instant. These steps are contained in the computer memory and are output by the program at each sampling instant. This method has the advantage of not being affected by any noise in the loop, but the effect of plant parameter changes may be greater due to the lack of feedback during the N sampling intervals. This controller was also tested and the results are given in the chapter on controller tests.

3.4 Stepper Motor Control Implementation

The stepper motor is a useful device in digital controllers. With each pulse the stepper motor advances 1 step. Since the motor does not move in case of power failure, it can be advantageous to use a stepper motor for the plant actuating signal. The assumption made is of course that the failure mode will not result in pulses to the motor. If the controller fails the inputs will remain at their last position, and unless the plant contains a free integrator it will not drift far from its operating point. It is also necessary to provide a direction signal for the motor in addition to the drive signal.

The stepper motor used here is a PHILIPS model K82701-P2, having a 7.5° step and a maximum speed of 200 steps per second. The interface from the controller to the stepper motor is simplified by the use of a PHILIPS stepper motor driver IC, SAA1027. This IC has pulse and direction inputs, and outputs to control the motor.

There are several ways to use stepper motors for control. One common method is the position control used in x-y plotters etc. The position is fed back from a potentiometer and compared with the desired position. The difference is the error signal and it is used to send pulses to the motor. The larger the error the higher the pulse rate.

The stepper motor can also be viewed as an integrator and thus can be used to perform the D/A conversion for plant control. One advantage as mentioned before is that the control value is held in case of controller failure, or power failure, unless, of course, the failure results in continuous pulses to the motor. The control output is changed by sending a number of pulses to the motor, as opposed to the D/A where a new control value is simply written into a register. This limits the speed of the control response.

In this mode the stepper motor is used as an open loop device and the pulse rate must be limited so that the motor has sufficient time to respond. To guarantee zero steady-state error in case pulses are missed some integral control must be used[3]. Pulses can be missed by the motor in case the load on the motor shaft exceeds the specified maximum torque, or when the pulse rate exceeds the maximum recommended value.

3.4.1 Positional Stepper Motor Control

This stepper motor controller measures the input voltage and translates it to a frequency according to figure 7. There is a dead zone in the controller with a width of approximately 1 step. This is necessary in order to prevent the motor from hunting around the set-point. The polarity of

---

[3] Janos Gertler, "Position vs Velocity Algorithms in d.d.c.", Instrument Practice, Dec. 1968, p. 1015

the input voltage also determines the direction control.

Error (Volts)	Frequency (Hz)
0.05	1
0.075	1.5
0.1	2
0.2	4
0.3	6
0.4	8
0.5	10
0.6	12
0.7	14
0.8	16
0.9	18
1.0	20
1.1	22
1.2	24
1.3	26
1.4	28
1.5	30
2.0	40
2.5	50
3.0	60
3.5	70
4.0	80
4.5	90
5.0	100
5.5	110
6.0	120
7.0	140
8.0	160
9.0	180
10.0	200

Fig. 7 Error vs Frequency for Stepper Control

3.4.2 Velocity Stepper Motor Control

In this control mode the previous PID output is subtracted from the present one to generate the velocity or rate control. This control value is then multiplied by an

overall gain constant for the stepper motor system in use and the resulting number is output as a series of pulses to the motor. The maximum number of pulses is limited to 255. The pulse rate is user selected and can be any even integer from 6 ms to 256 ms.

4. Controller Design

This chapter deals with the design of various types of controllers.

4.1 PID design

The design of PID controllers is a subject well covered in numerous publications. See, for example, Ogata[4]. As long as the sampling period is short compared to the time constants of the plant, the controller coefficients can be designed using standard methods found in the texts. A controller can also be found experimentally by simply varying the coefficients until satisfactory control is reached.

Since the exact plant transfer function is often not known in industrial processes, approximation techniques are commonly used for controller design. The Ziegler-Nichols method[5] is well known and uses two easily performed measurements to determine the controller coefficients. The necessary measurements are the time lag and response rate of the system. They are measured by applying a step to the open-loop plant and measuring the output response. Some fine-tuning of the controller is also often required.

---

[4] Katsuhiko Ogata, "Modern Control Engineering", Prentice Hall, New Jersey, 1970

[5] Ziegler, J.G. and Nichols, N.B., "Optimum Settings for Automatic Controllers", ASME Transactions, 64, 1942, p.759

In general,

1) The proportional gain is increased to speed up response and lower offset error. There is a limit beyond which the system will be unstable.
2) Integral control is added to reduce the steady-state error if the plant does not contain a free integrator. Integral control introduces phase shift which can cause instability.
3) Derivative control can be used to increase the stability of a system with a lagging phase margin. i.e. too many integrators. Derivative control increases the noise sensitivity of the system.

The important difference, in using the digital PID controller, is when the sampling time becomes comparable to the plant time constants. A system stable for fast sampling times will exhibit an increasing overshoot as the sampling time is increased. At some point the system will become unstable. This problem is related to the phase shift introduced when a signal is reconstructed by a zero-order hold device. Excessive phase shift will cause instability due to the loss of sufficient phase margin.

4.2 Dead-Beat Design

A dead-beat controller is one in which the output variable reaches the set-point in the shortest time possible, and the derivatives of the output reach zero at

the same time. If the control output is not constrained in magnitude it can be shown that the output will reach the set-point in a minimum of N sampling periods. N is the order of the system. If the control is constrained, more than N samples will be required. The magnitude of the control outputs are related to the sampling time, so that the control magnitudes can be reduced by increasing the sampling period.

Dead-beat response can be accomplished by designing the transfer function to cancel all poles and zeros of the combined system response with the exception of the single pole at z=1. However, this method allows intersample ripple in the output of the plant. This can be eliminated if the controller is designed such that all derivatives are equal to zero and the output is equal to the set-point after the required number of samples. This is the case in which we are primarily interested.

If the plant contains a free integrator, the output of the controller is zero at the end of the N samples. If, however, the plant does not contain a free integrator, then the D(z) controller must maintain an output. i.e. the digital transfer function must contain the integrator.

4.2.1 Dead-Beat Controller Pulse Transfer Function Derivation

It is convenient to use the state variable representation. The system transfer function is given by,

$$\frac{d}{dt} x = F x + G u$$

and the output is,

$$y = C x$$

This is shown in the block diagram of figure 8.

It is possible to design the dead-beat controller in general[6]. The resulting D(z) function is given by,

$$D(z) = \left[\sum_{k=0}^{N-1} z^{-k} P(k) + P(N) \frac{z^{-N}}{1-z^{-1}}\right] \left[\sum_{k=0}^{N-1} z^{-k} \left(I - \sum_{\ell=0}^{N-1} \{CA^{k-\ell-1} BP(\ell)\}\right)\right]^{-1}$$

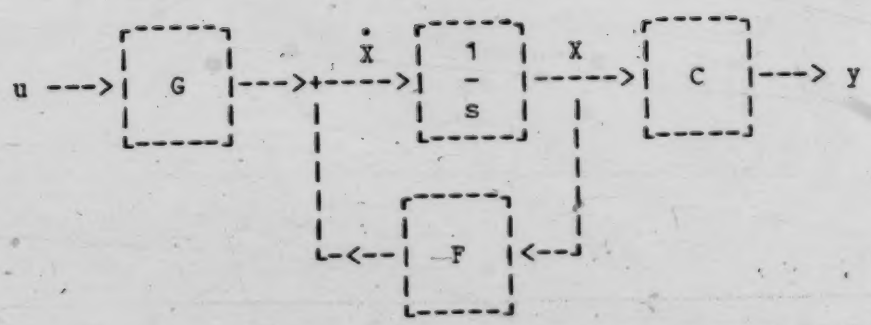


Fig. 8 State Variable Transfer Function

---

[6] J. A. Cadzow and H. R. Martens, "Discrete-Time and Computer Control Systems", 1970, pp. 259-263

Where P is found from,

$$\begin{bmatrix} CA^{N-1}B & CA^{N-2}B & \cdots & CB & 0 \\ FA^{N-1}B & FA^{N-2}B & \cdots & FB & G \end{bmatrix} \begin{bmatrix} e_2(0) \\ e_2(1) \\ \cdot \\ \cdot \\ \cdot \\ e_2(N-1) \\ e_2(N) \end{bmatrix} = \begin{bmatrix} x_0 \\ 0 \end{bmatrix}$$

4.2.2 Computer Program to Calculate Dead-Beat

The above given algorithm for the design of the minimal time response controller can easily be programmed on a large digital computer. Since the solutions are in matrix form, a particularly convenient language to use is APL. APL has all of the matrix manipulation operators as primitives.

A program was developed in APL to calculate the dead-beat coefficients for second and third order systems. The program is general and the systems are not restricted in requiring the presence of a free integrator. Note however, that the program is restricted to systems having one input and one output, as is the controller system under discussion. The discrete state matrix is given by,

$$x(k+1) = A\,x(k) + B\,u(k)$$

and,

$$y = C\,x(k).$$

The program requires the F matrix, the G matrix, the output matrix C, and the sampling period T. The results are the A matrix, the B matrix, and the D(z) coefficients. The coefficients are output in the same form as the normalized

D(z) form described earlier.

The program is run by the following sequence.

1. Signon to MTS
2. $RUN *APL
3.)LOAD DEAD
4. DEAD
5. The program then prompts for the needed matrices, which are entered by rows, with each row going from left to right
6. When the calculations are completed the program prints the results

The algorithm used is the series evaluation of the state transition matrix given in Cadzow and Martens[7]. The program uses 100 terms of the series which results in at least 8 digits of accuracy.

4.3 Dual Mode Design

In general, the basic design procedure for the dual-mode type controller is as follows.

1) Design the PID controller using standard design techniques, using sampling time T0
2) Design the dead-beat response considering the restraints of control magnitude and the desired step response time, using sampling time T1
3) Enter the coefficients into the controller and test the system, noting that either the D(z) or

---

[7] loc cit, pp. 389-390

the table look-up may be used

4.4 Digital Filter Design

There are many advantages to implementing filters with digital techniques. Some of these are;

1) Repeatable performance from unit to unit
2) Precision limited only by arithmetic capabilities and D/A, A/D resolution limits
3) Digital domain has no impedance matching difficulties
4) Critical break frequencies can be placed with no tuning or adjustments required
5) No dependence on component value variation due to temperature or age
6) Filter characteristics are easily modified by changing digital words in memory

Since there is a wealth of information on analog filter design, it can be advantageous to design the filter using analog methods, and then transform the filter to the digital form. For analog filter design see 'Simplified Modern Filter Design'. [8]

One method known as the impulse invariant transform [9] uses the following steps.

1) Design the filter with analog techniques

---

[8] Geffe, P.R., "Simplified Modern Filter Design", John F. Rider, New York, 1963
[9] Childers and Durling, "Digital Filtering and Signal Processing", 1975

2) Take the z-transform of the s-domain impulse transfer function

3) Implement the filter using the controllers D(z) operational mode

As a simple illustration of this method take the transfer function,

$$H(s) = \frac{1}{s + 31.4}$$

This corresponds to the low-pass filter with the breakpoint at 5 Hz. Taking the z-transform of the above with a sample time of 10 ms yields,

$$H(z) = \frac{1}{1 - 0.730\ z^{-1}}$$

This filter was implemented and tested on the system.

Note the relationships between the transfer functions of the digital and analog filters. In the s-domain s^{-1} represents analog storage, an integrator. In the z-domain the z^{-1} operator represents time storage, or delay.

The z-transform technique of digital filter design is covered in numerous texts. An excellent reference source is 'Digital Filtering'. [10]

The frequency range that can be covered with the controller described in this thesis is limited by the

---

[10] loc cit

sampling rate, which in turn is limited by the CPU calculating time. By Shannon's sampling law the input signal must be sampled twice every period. Since the fastest sampling rate is 10 ms, the upper frequency limit will be 50 Hz.

5. Controller Tests

This chapter describes various tests performed with the controller. Pictures are included showing the response of the system and the controller.

The following photographs illustrating the controller have the top trace showing the plant output and the bottom trace showing the controller output. Unless stated otherwise the photographs have,

1) Sweep time 2 s/div
2) Top trace 2 V/div
3) Bottom trace 5 V/div

For most systems a 4 volt step was applied from the keyboard.

Figure 9 lists the plant transfer functions that were used for the controller tests. Also listed is the dead-beat controller, for which the coefficients were calculated by running the APL program described earlier.

System	G(s)	Sample Time	D(z) for Dead-Beat
A	$\frac{1}{s+1}$	1 sec	$\frac{1.58 - 0.582z^{-1}}{1 - z^{-1}}$
B	$\frac{1}{s(s+1)}$	1 sec	$\frac{1.58 - 0.582z^{-1}}{1 + 0.418z^{-1}}$
C	$\frac{5}{(s+1)(s+2)}$	0.693 s	$\frac{1.07 - 0.8z^{-1} + 0.133z^{-2}}{1 - 0.667z^{-1} - 0.333z^{-2}}$
D	$\frac{5}{s(s+1)(s+5)}$	0.693 s	$\frac{1.56 - 1.19z^{-1} + 0.191^{-2}}{1 + 0.735z^{-1} + 0.085z^{-2}}$
E	$\frac{1}{s^2 + s + 1}$		

Fig. 9 Plant Transfer Functions

5.1 PID Mode Tests

The proportional part of the PID controller was tested with a known input and gain constant Kp. The inputs and Kp were changed and the output checked for the proper value.

To test the integral part a step was applied to the system with only the integral gain constant non-zero. The results of this test for two different sampling periods are shown in figures 10 and 11.

The derivative part was tested similarly as seen in

figures 12 and 13.

The results of the above confirm the operation of the PID control elements.

A simple PI controller was tested by implementing the control function Kp=1, Ki=0.1, and T=0.1 seconds using system A. The response of this system seen in figure 14 can be compared to the response of the dead beat controller for the same system, figure 19.

System B was controlled with Kp=1 and sampling times varied from 0.1 to 4 seconds to illustrate an analog stable system becoming unstable as the sampling time is increased. This is seen in figures 15 to 18. The instability is clearly seen for the sample time of 4 seconds.

The applied step was 4 volts except in the last case where it was 5 volts. These four illustrations use a time base of 5 s/div. Note the scale change of the last case , 5 V/div and 10 V/div for the top and bottom trace respectively.

5.2 D(z) Mode Tests

Dead-beat controllers were designed for systems A, C, and D of figure 9. Figure 20 shows the control of a plant with a free integrator. Plants without free integrators are shown in figures 19 and 21. A 5 volt step was appled to system A and 4 volt steps to the others. Dead-beat response was achieved in all cases.

5.3 Dual-Mode Tests

System B was used for the dual-mode control operation. For all cases the step is 4 volts. The proportional controller has Kp=1 and T=0.1 s. The dead-beat controller uses T=1 s. The dual-mode in normal operation is seen in figure 23.

5.3.1 Test with Plant Pole Variation

It is interesting to compare the table look-up method with the D(z) dead-beat method when the plant parameters are varied. First the pole at s = -1 is moved to -1.25, seen in figures 24 and 25. Next the pole is moved to -0.75, seen in figures 26 and 27.

It is seen that the D(z) control is more accurate than the table method when the plant changes. This is expected since the table method is essentially open loop control for the dead-beat period. The D(z) is closed loop and will tend to compensate for the parameter change.

5.3.2 Test with Noise Added to Loop

A similar test was performed with the two dual-mode controller types in the presence of noise. The noise was injected into the loop from a SERVOMEX noise generator. The noise bandwidth is from 0.05 to 50 Hz with an RMS value of 0.25 volts.

As can be seen in figures 29 and 28 the table method is more accurate than the D(z). This is also to be expected due

to the nature of the D(z) transfer function. The output values are calculated from instantaneous input samples and any noise in the loop will effect this measurement, causing an incorrect output. The table method does not measure the input and is therefore not affected by the presence of noise.

The dual-mode controller was also tested with a plant without a free integrator. This confirms the transition between PI and dead-beat. The PI control was T=0.1, Kp=1, and Ki=0.1. The result can be seen in figure 22.

5.4 Stepper Motor Control Tests

The stepper motor was connected to a potentiometer through a 5 to 1 gear reduction. The potentiometer was connected to ±12 volts.

5.4.1 Positional Stepper Motor Test

The wiper voltage was used for the feedback signal, and was connected to the controller input. Figure 30 shows the response of this system to a 5 volt step. The top trace is 2 V/div and the bottom trace is 5 V/div. The sweep rate is 100 ms/div.

5.4.2 Velocity Algorithm Stepper Motor Test

First the output of the three control elements was tested, as shown in figures 31, 32, and 33. All of these have;

1) Top trace = 5 V/div

2) Bottom trace = 2 V/div

3) Sample time = 2 Sec

4) Pulse time = 10 msec

5) Sweep rate = 2 Sec/div

6) Velocity overall gain = 0.095

The controller gains are;

1) Kp = 0.5

2) Ki = 0.1

3) Kd = 1.0

The individual response of the PID elements is seen to be correct.

To further test the stepper motor control system, a PID controller for system E was implemented. System E was time scaled by a factor of 10 to reduce the effect of the slow motor response. The following data holds for figure 34.

1) Vertical 2 volts and 5 volts /div

2) Horizontal 15 sec/div

3) Overall gain = 0.06

4) Sample time = 2 sec

5) Step pulse time = 10 msec

6) Applied step = 4 volts

7) Kp = 2.00

8) Ki = 0.150

9) Kd = 1.50

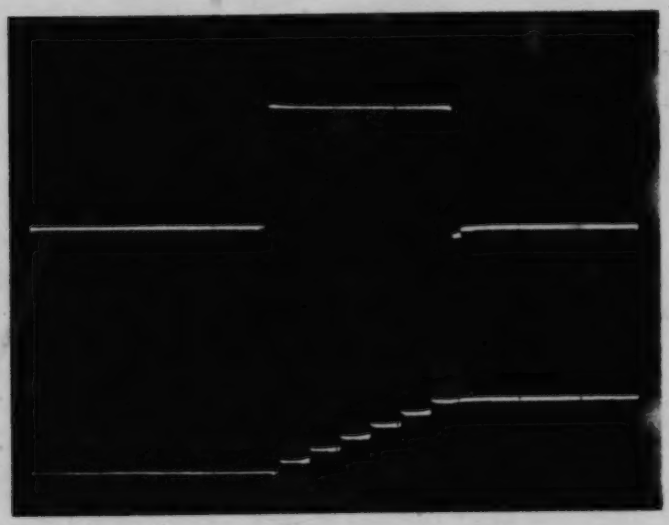


Fig. 10 Integrator Ki=0.1 T=1 S

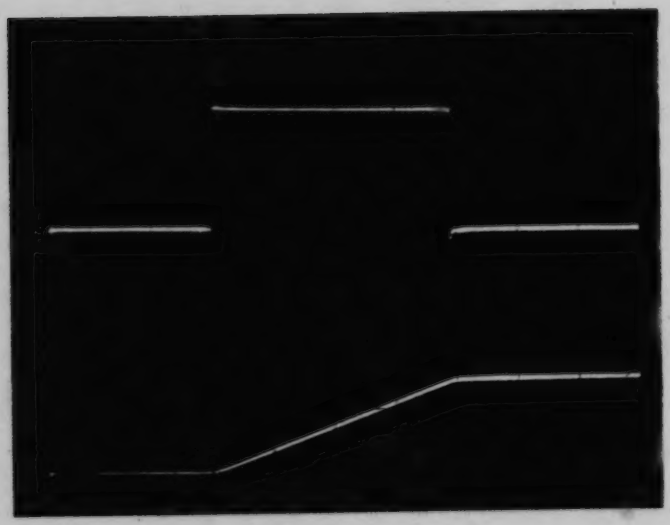


Fig. 11 Integrator Ki=0.001 T=10 mS

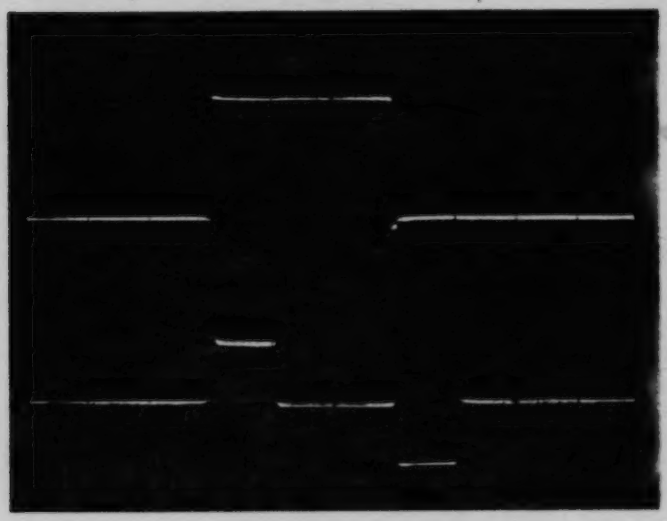


Fig. 12 Differentiator Kd=2 T=0.5 S

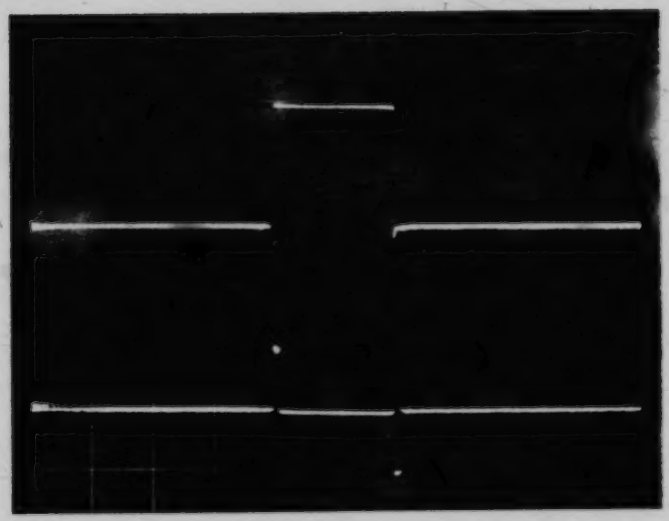


Fig. 13 Differentiator Kd=2 T=50 mS

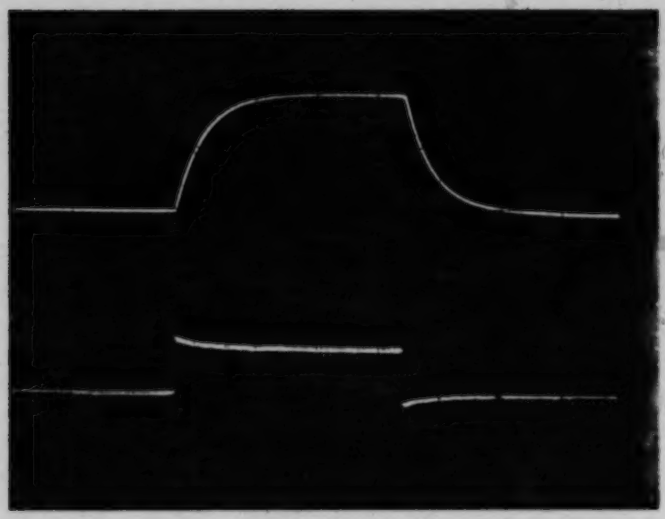


Fig. 14 P-I Control of System A

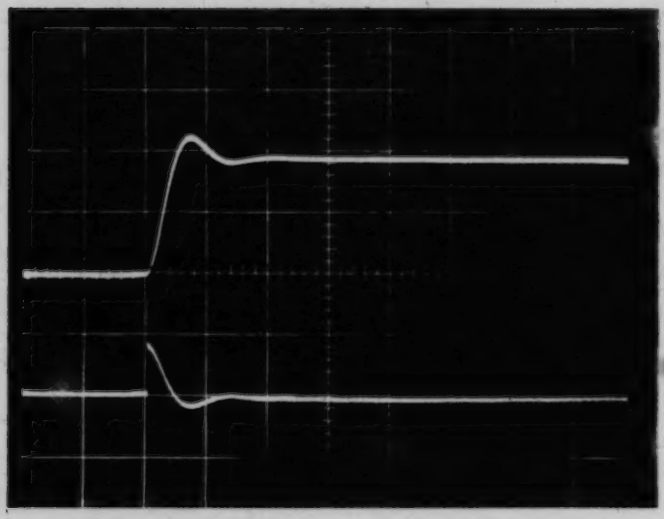


Fig. 15 System A Kp=1 T=0.1 S

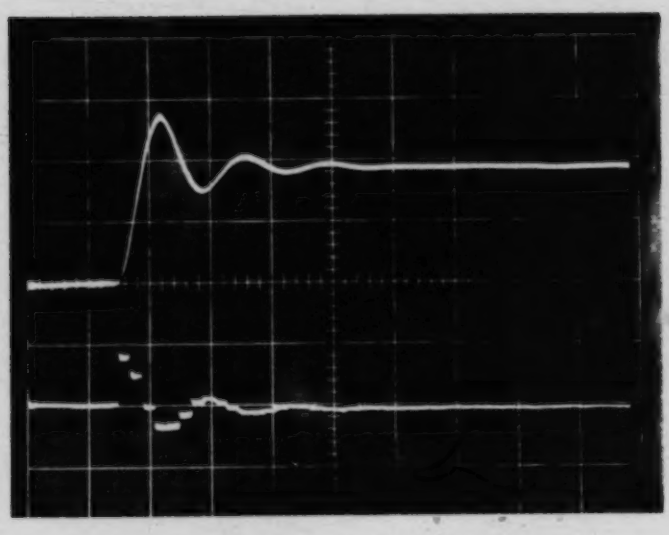


Fig. 16 System A Kp=1 T=1 S

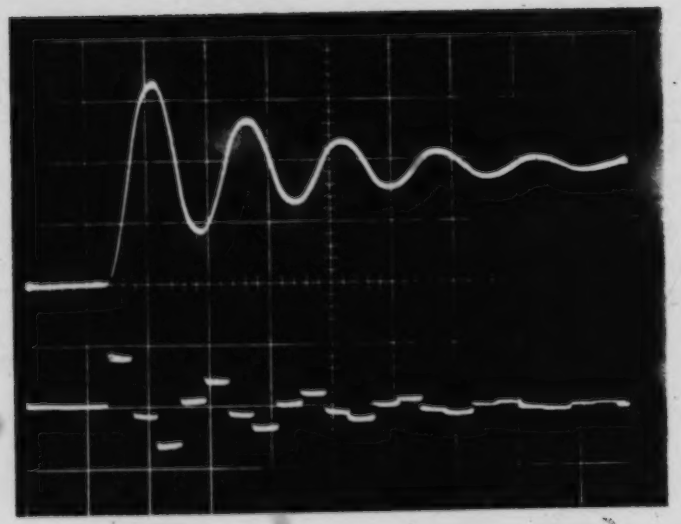


Fig. 17 System A Kp=1 T=2 S

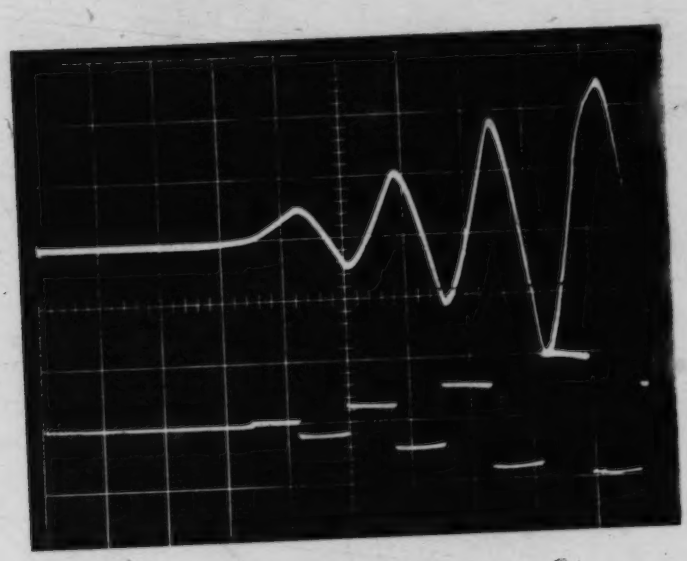


Fig. 18 System A Kp=1 T=4 S

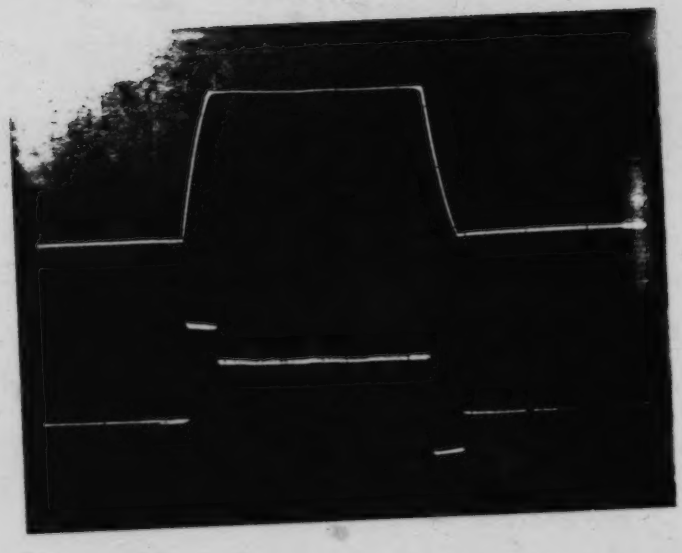


Fig. 19 Dead-Beat Control of System A

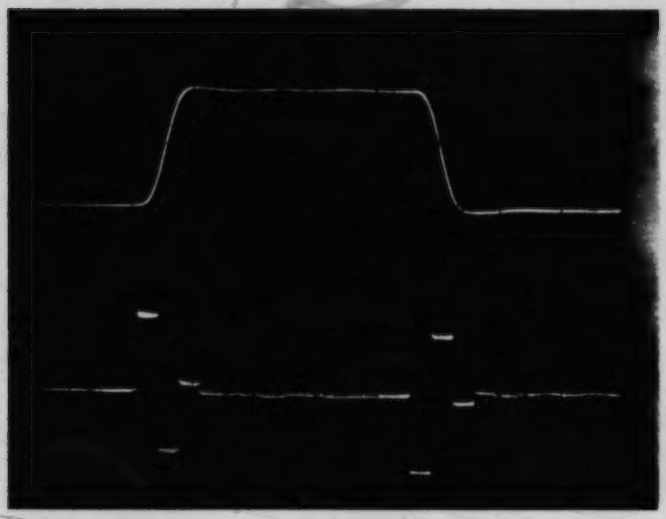


Fig. 20 Dead-Beat Ccntrol of System D

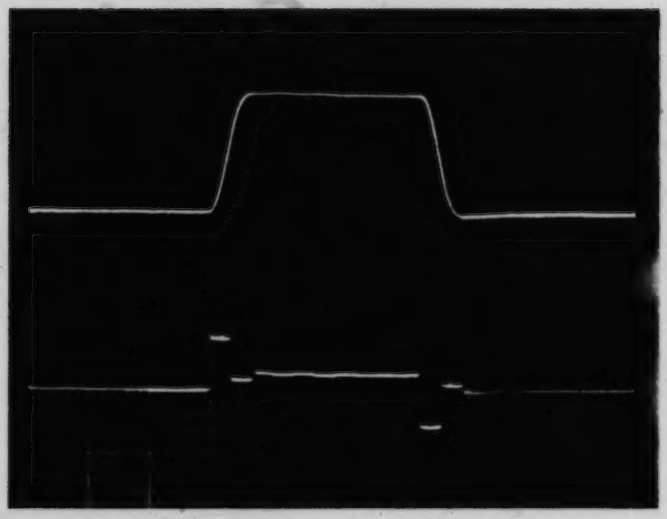


Fig. 21 Dead-Beat Control of System C

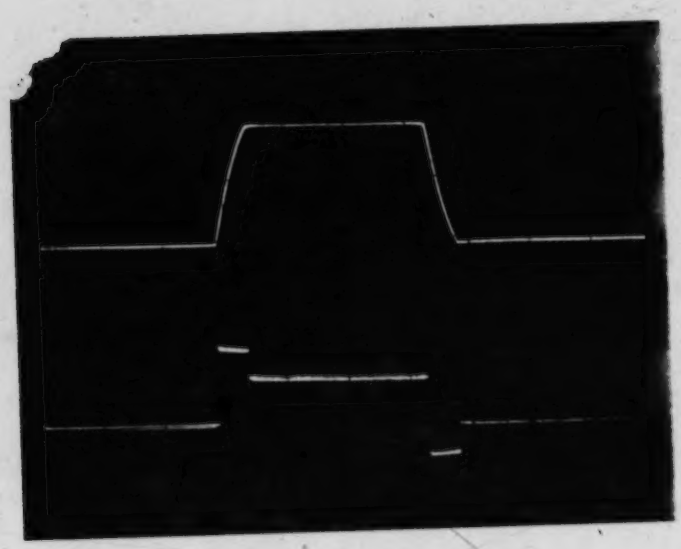


Fig. 22 Dual-Mode Control of System A

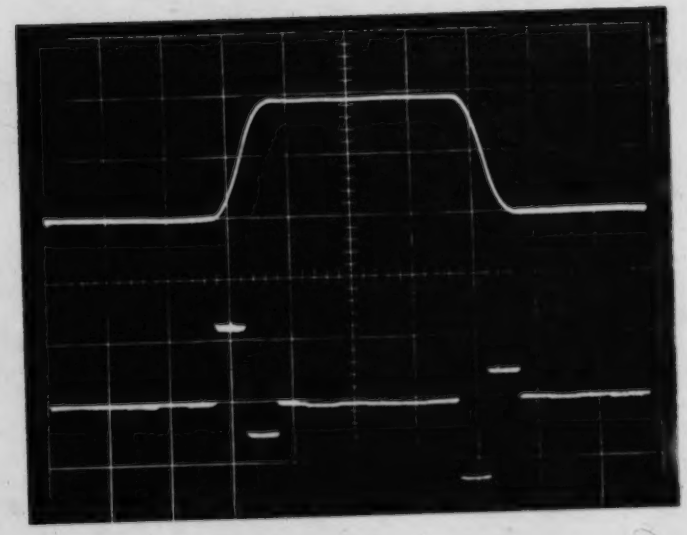


Fig. 23 Dual-Mode Control of System B

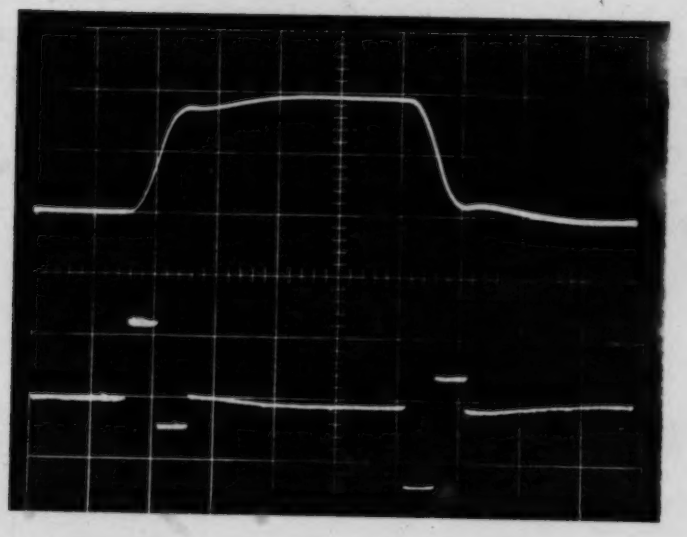


Fig. 24 D(z) Dual-Mode with pole at -1.25

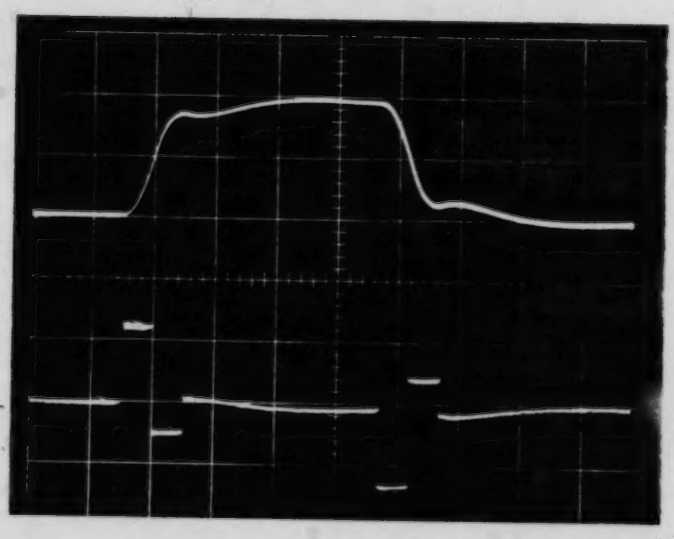


Fig. 25 Table Dual-Mode with pole at -1.25

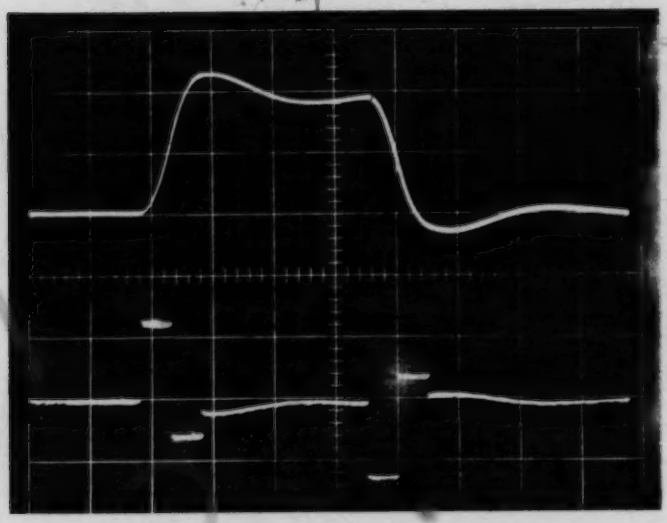


Fig. 26 D(z) Dual-Mode with pole at -0.75

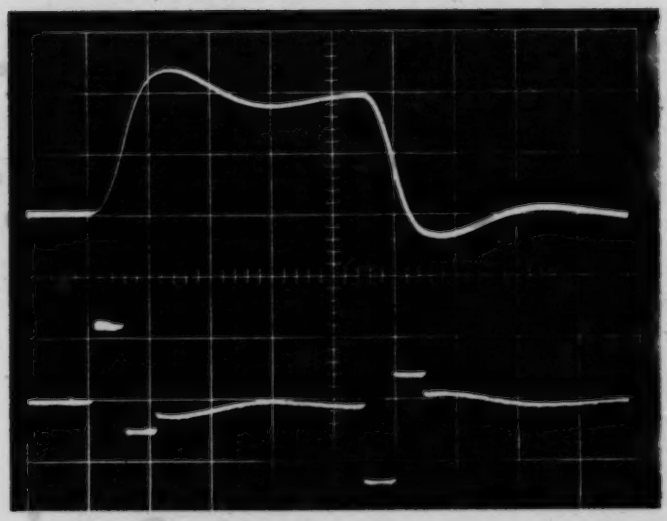


Fig. 27 Table Dual-Mode with pole at -0.75

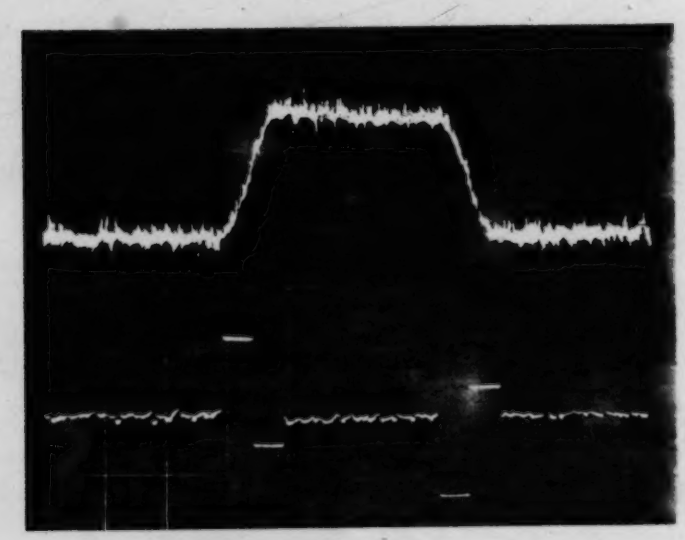


Fig. 28 Table Dual-Mode Control of B with Noise

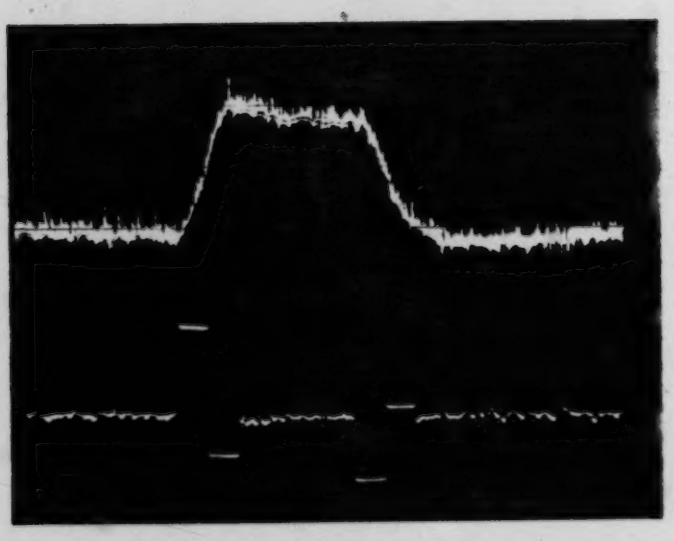


Fig. 29 D(z) Dual-Mode Control of B with Noise

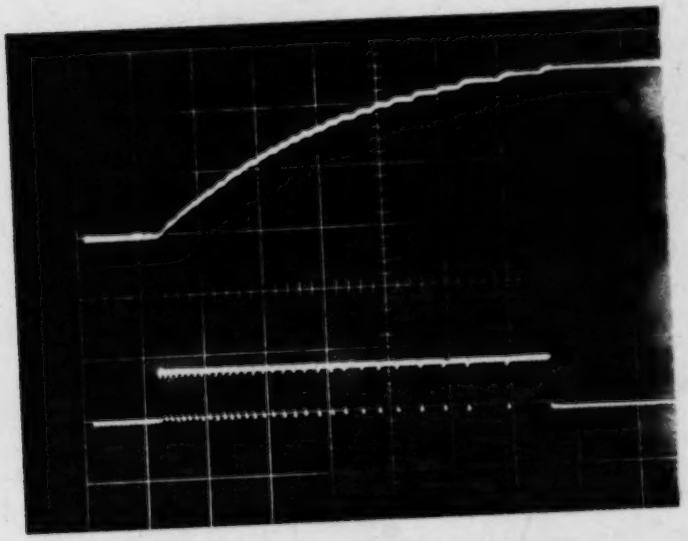


Fig. 30 Stepper Motor Positional Response

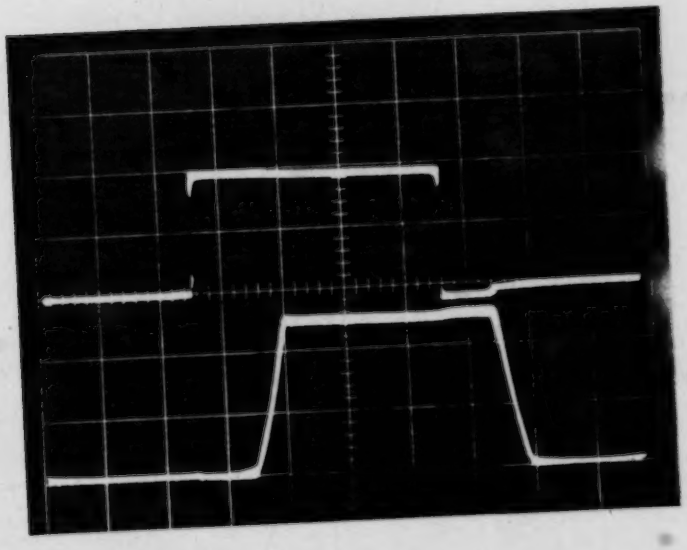


Fig. 31 Velocity Algorithm, Proportional Output

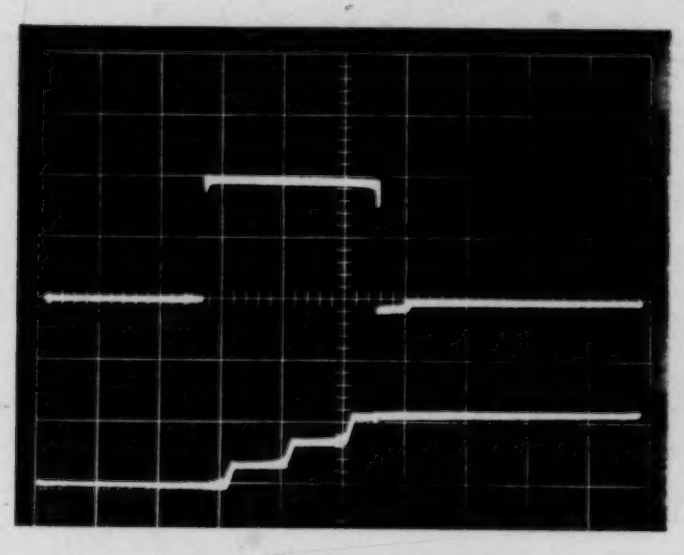


Fig. 32 Velocity Algorithm, Integral Output

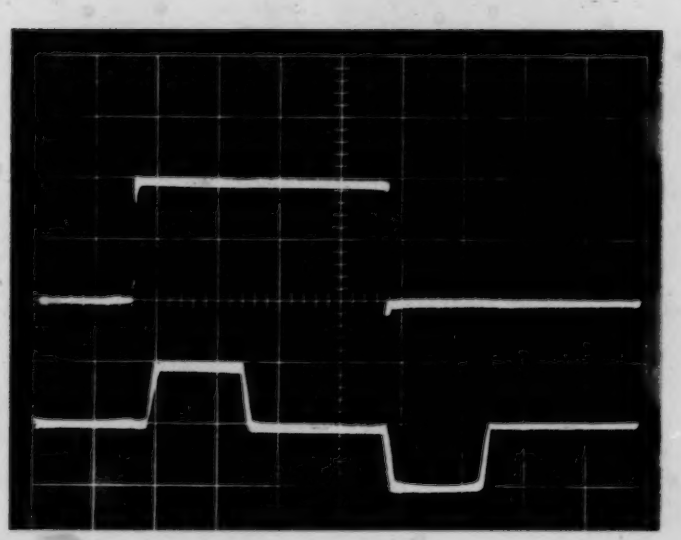


Fig. 33 Velocity Algorithm, Derivative Output

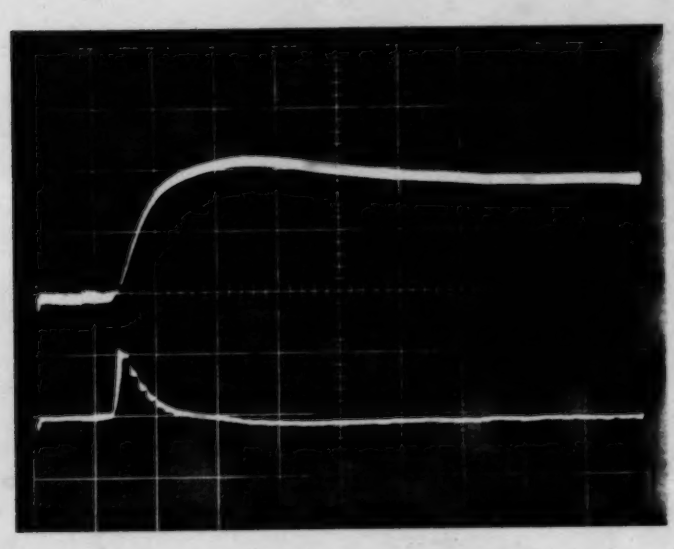


Fig. 34 Velocity Algorithm, PI Control

Conclusions

User experience has shown that this micro-processor controller system is useful and convenient to use in the laboratory. Combined with the plant G(s) simulated on an analog computer it provides a simple means to study the z-transform D(z) and the digital PID control algorithm. It also allows the combined dead-beat and PID to be studied. The coefficients are easily changed from the keyboard and the effects can be seen immediately.

Future Plans

During the next year the device will be used in the undergraduate controls laboratory. Student experience will then provide evaluation of the device in actual student laboratory practice practice, and perhaps suggest further improvements.

The device will also be used as a digital filter for signal conditioning of certain biological signals.

There are many other extensions possible to this machine. Several under consideration are given below.

1) Word size truncation to show the effect of fewer bits on the control accuracy
2) A proportional-integral bang-bang controller [11]
3) Selectable time delay before output of control

---

[11] L.M. Zoss, K. Mortimer, and K.H. Vetter,"Novel Bang-Bang Algorithm for Direct Digital Control", Instrumentation Technology, April 1970

Bibliography

1. Martens, "IEEE Transactions on Education", Vol. E-11, No. 4, Sept. 1968
2. B. J. Leon and S. C. Bass, "Designers Guide to: Digital Filters", Electronic Design, June 20, 1974
3. Katsuhiki Ogata, "Modern Control Engineering", Prentice Hall, New Jersey, 1970
4. J. A. Cadzow and H. R. Martens, "Discrete-Time and Computer Control Systems", 1970, pp. 259-263
5. Childers and Durling, "Digital Filtering and Signal Processing", 1975
6. B. C. Kuo, "Discrete-Data Control Systems", 1974

Appendix

The appendix contains system flowcharts, the memory map, schematic diagrams, the program source, and the operators manual.

Operators Manual

This laboratory device is a microprocessor based digital controller.

The unit has two major types of feedback control, the standard PID algorithm and the D(z) control, which can be used for deadbeat controllers or digital filters. The control mode, sampling time, and the various coefficients are all conveniently entered from the keyboard. A diagram of the keyboard is shown in figure 35.

The digital controller features sampling times from 10 ms to 30 s. The A/D and D/A converters are capable of analog voltages in the range of ±10 volts with a resolution of 12 bits.

RST	LDZ	LCM	7	8	9
CLI	DDZ	DCM	4	5	6
LSP	LST	LDC	1	2	3
DSP	DST	DSC	0	-	CLR

Fig. 35 Controller Keyboard

Number Buttons

The right half of the keyboard has the numerical keys, the minus sign, and the clear command. The clear button is used to clear the display, while the numbered keys and the minus sign are used in entering numbers into the display. Numbers from the display are entered into memory by pressing the load set-point, load sampling period, or load coefficient keys, after the digital display shows the desired number. To enter a coefficient the number keys are pressed until the display shows the desired number in normalized scientific notation, (fractional mantissa followed by exponent), after which one of the load function keys is pressed, followed in some cases by another number, the subscript value. i.e. (LDC,n) where n is the subscript and refers to the specific coefficient being stored. Both the sampling periods, set-point, and coefficients have limited ranges, and attempting to load numbers outside of these ranges will generate error messages, and the number will not be stored. The display automatically adds the decimal point at the first position when the first digit is pressed. Entering single numbers for D(z) order or control mode is done by pressing the function key followed by the single digit number.

The left half of the keyboard contains all of the function buttons. Their operation is described in the following section.

Function Buttons

RST

This is the button to reset the microprocessor. It will clear all memory locations. This button should only be used if the processor has gotten lost in its program.

CLI

This key clears the integrator used by the PID control mode. This is the same as loading the integrator with an initial value of zero.

LDZ

This function is used to change the order of the D(z) controller. LDZ is pressed followed by the desired order for the D(z) transfer function. The controller will accept orders from 1 to 3.

DDZ

Pressing DDZ will cause the D(z) order to be displayed.

LCM

This button is used to select the control mode. LCM is pressed followed by a control mode from 0 to 7. The following control modes are implemented:

0-PID

This is the standard PID controller. This mode implements the discrete PID algorithm and uses coefficients Kp, Ki, and Kd. (See LDC for loading) The three parts of the control are implemented according to the equations given in the thesis body. The PID controller uses sample time T0. E1 is the sampled input (i.e. the plant output) and E2 is the

controller output. Note that entering a set-point change causes the derivative control element to be disabled for four sampling periods.

1-D(z)

This is a D(z) controller which is intended to be used in the same configuration as the PID controller. The control algorithm implemented is discussed in the thesis body under D(z) Implementation.

E2 is the output signal and E1 is the sampled error signal. The coefficients used by this control mode are A0, A1, A2, A3, B1, B2, and B3. signal. The sampling time used is T0. The maximum order (N) is 3. This control mode is used to implement various digital controllers, of which a dead-beat or minimal time response is one possibility. (see LST button)

2-Dual Mode D(z)

This control mode uses the PID mode as long as no set-point changes are initiated. If a set-point change is entered the controller switches over to the D(z) mode for N samples, where N is the order of the D(z) controller. The D(z) function should be set up for dead-beat response. The PID controller in this case uses sampling time T0 while the D(z) uses sampling time T1. The use of separate sample times allows for flexibility. The advantage of this mode is to have the noise immunity of the PID controller combined with the ability of the dead-beat response to perform state changes of a plant in minimal time.

3-Dual Mode with Table Look-up

This mode is the same as mode 2 with the exception that instead of switching over to the D(z) mode for set-point changes, the controller will output to E2 values that are held in memory. The control table values have been previously calculated for the known plant transfer function G(s) and are stored in coefficient locations A0, A1, A2, and A3. If the D(z) order is 1 A0 followed by A1 will be output, multiplied by the error, before the controller switches back to the PID mode. If the order is 3 all of the values from A0 to A3 will be output sequentially with the time between each given by sampling time T1.

4-Stepper Motor Velocity Algorithm Control

This mode has pulse and direction outputs intended for stepper motor control. At each sampling instant the A/D is read and the error signal generated by subtracting from the set-point. Next the PID controller is called to calculate the control signal. The control signal is then multiplied by a gain constant SG (in location B3) to give an integer result. This integer is truncated to 8 bits and stored in a counter. The counter then outputs the count at a pulse rate which is stored in location T2, (LST,2). The gain constant is related to the number of steps per volt for the motor/system under control.

Used in this manner the stepper motor can be viewed as a digital to analog converter. It has the advantage of holding its output in case of a power failure.

When the system is set up the stepper motor should be manually positioned at its zero control position and CLI pressed to clear the integrator before power up of the motor. Integral control should always be used due to the possibility of the motor missing pulses under high load and fast pulse conditions.

Note also that sampling periods are limited by the response time of the motor. At a step pulse rate of 10 ms, 255 pulses (the maximum) will take 2.55 Seconds.

5-Stepper Motor Positional Control

This mode uses the pulse and direction outputs the same as mode 4. The PID elements are also used but often only Kp would be non-zero. This control mode uses the error signal to generate a pulse rate to the motor. A larger error causes a higher frequency. The frequency varies from 1 Hz to 200 Hz. A complete table is found in the thesis body.

This control mode is a common method used in x-y plotters etc.

DCM

This key is used to display the selected control mode on the digital display.

LSP The set-point is loaded using this function button. When the display contains the desired value for the set-point the button is pressed and the set-point will be loaded. This function also initiates certain other control actions in

control mode 0, 2, and 3. In mode 0 the derivative control is disabled for 4 samples. This is the bumpless transfer feature and eliminates large control values, due to the derivative term, when the set-point is changed. In modes 2 and 3 a set-point change initiates the switch over to dead-beat control for N + 1 sample periods (N is the D(z) order).

Since the D/A converter range is ±10 volts, set-points cannot be loaded outside of these limits. The D/A converter has 12 bits of resolution resulting in the minimum step size of 4.88 mV. Attempting to load a set-point of magnitude less than 3.4 mV (with the exception of 0) will result in an error message on the display. Loading a set-point of 0 is of course permitted.

DSP

This function button causes the current set-point to be shown on the digital display.

LST

When the display shows the desired sample time in seconds, (as usual in normalized scientific notation), this function button is pressed followed by 0, 1, or 2 to load sampling time T0, T1, or T2 respectively. T0 is used by all control modes and T1 is additionally used by modes 2 and 3.

The range of sampling periods allowed is 10 ms to 30 s, and values outside of this range will not be stored.

T2 is the pulse rate used to control the stepper motor in mode 4. The minimum value is 6 ms, and even periods can be loaded up to 254 ms. Odd numbers may be loaded but they will be rounded to even since the resolution of the hardware is 2 ms.

DST

DST is used to display the set sampling times. To use it the key is pressed followed by 0 or 1 to display T0 or T1 respectively.

LDC

This function is used to load the various coefficients used by the controller. Similarly to LST the display must first contain the desired number, when it does ,LDC is pressed followed by a digit from 0 to 9.

The first three are used by the PID controller.

0 - Kp the proportional gain

1 - Ki/T the integral gain divided by the sampling period

2 - Kd*T the derivative gain times the sampling period

The last seven are used by the D(z) controller.

3 - A0

4 - A1

5 - A2

6 - A3

7 - B1

8 - B2

9 - B3 or SG

SG is the overall gain for mode 4.

The range of coefficient magnitude which the controller will accept is 0.001E-3 to 0.999E+3. Numbers outside this range will generate an error message and the original coefficient will remain in storage.

ISC

This function button displays the coefficients stored in the controllers memory. Similarly to LDC it is pressed followed by a digit from 0 to 9.

Error Messages

Pressing keys in improper order, attempting to load numbers that are out of range, or using subscripts that are out of range will cause an error message to be displayed on the digital display.

The following error messages are defined.

Error 0 Subscript number is out of range.
e.g. (LST,3) would cause this error

Error 1 Coefficient magnitude is out of range.
e.g. display=.145E4 (LDC,2) gives this error

Error 2 Sample time is out of range.
e.g. display=.100E-2 (LST,0) causes

this error to be displayed

Error 3 Set-point number is out of range.
e.g. display=.100E3 ↵(LSP) gives
this error message

Error 4 D(z) order or control mode number
out of range.
e.g. (LCM,8) would give this error

Error 5 Another function was called before
finishing the first one.
e.g. (LDZ,LSP) would cause this error
since LDZ is expecting an integer.

Short Form Operators Manual

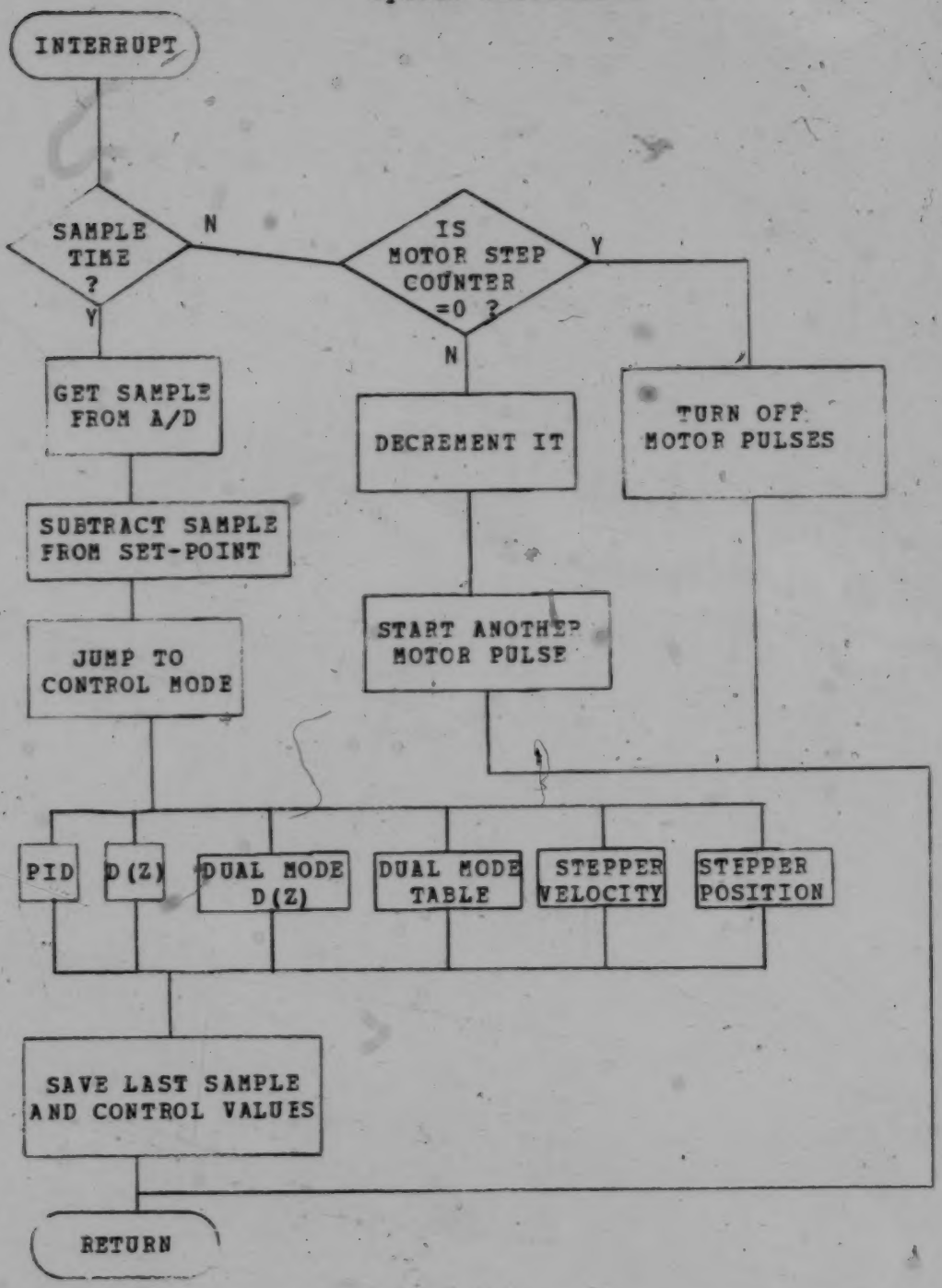
```
RST     Reset the microprocessor, clears the memory
CLI     Clear the PID integrator
LDZ,n   Set D(z) order to n
DDZ     Display D(z) order
LCM,n   Select control mode n
          Control modes are,
          0  PID
          1  D(z)
          2  Dual-mode with D(z) for step
          3  Dual-mode with table for step
          4  Stepper Motor Velocity Algorithm
          5  Stepper Motor Positional
DCM     Display the selected control mode
LSP     Load the set-point
DSP     Display the set-point
LST,n   Load sample time n
          All modes use time 0 and
          dual-mode controllers use
          time 1 for set-point changes
          Time 2 is the step motor
          pulse rate (mode 4 only)
DST,n   Display sample time n
LDC,n   Load coefficient n, n=0 to 9
          The coefficients are,
          0  Kp          5  A2
          1  Ki/T        6  A3
          2  Kd*T        7  B1
          3  A0          8  B2
          4  A1          9  B3 or SG
    SG is the stepper control gain (mode 4 only)
DSC,n   Display coefficient n


   The error messages are,
    Error 0  Subscript number is out of range.
    Error 1  Coefficient magnitude is out of range
    Error 2  Sample time is out of range
    Error 3  Set-point value is out of range
    Error 4  D(z) order or control mode number
             out of range
    Error 5  Another function was called before
             finishing the first one
```

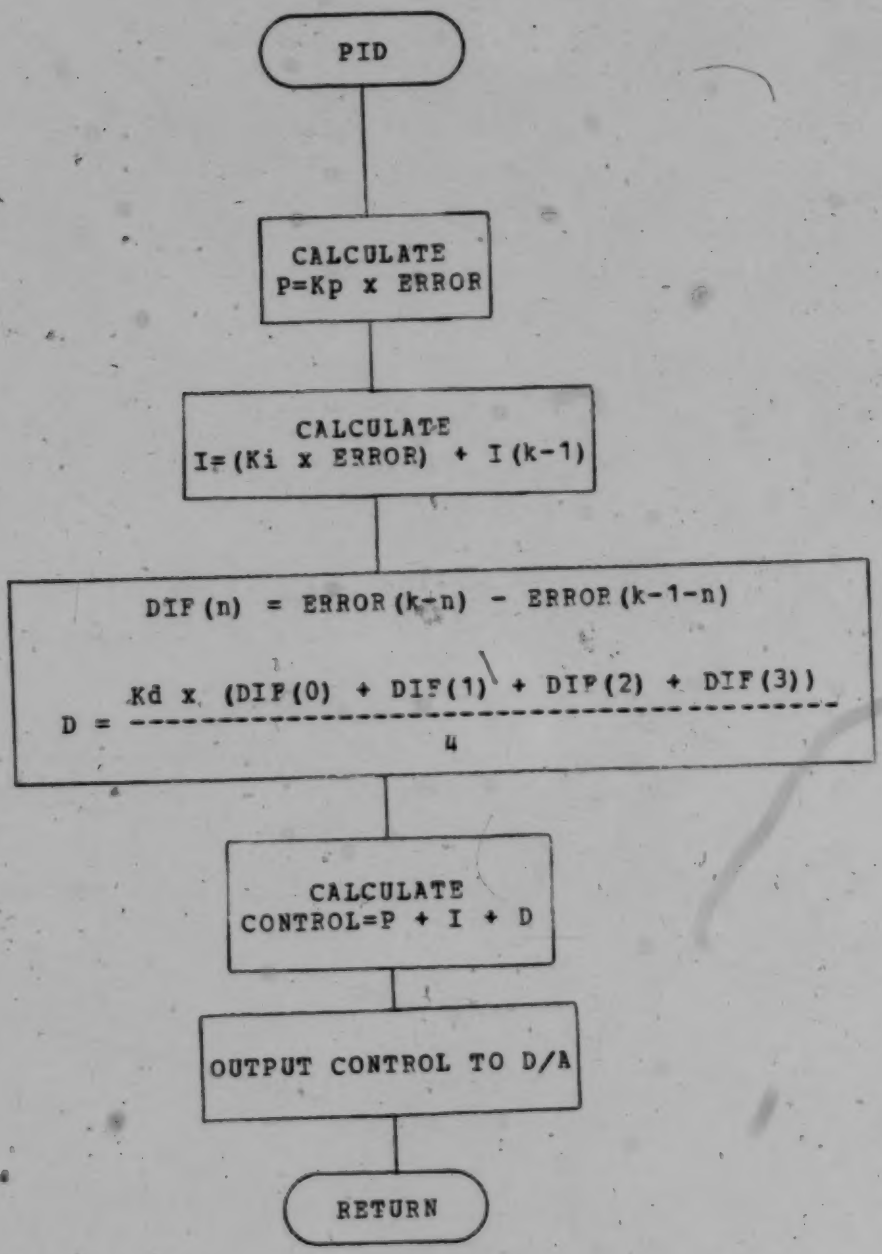
System Memory Map

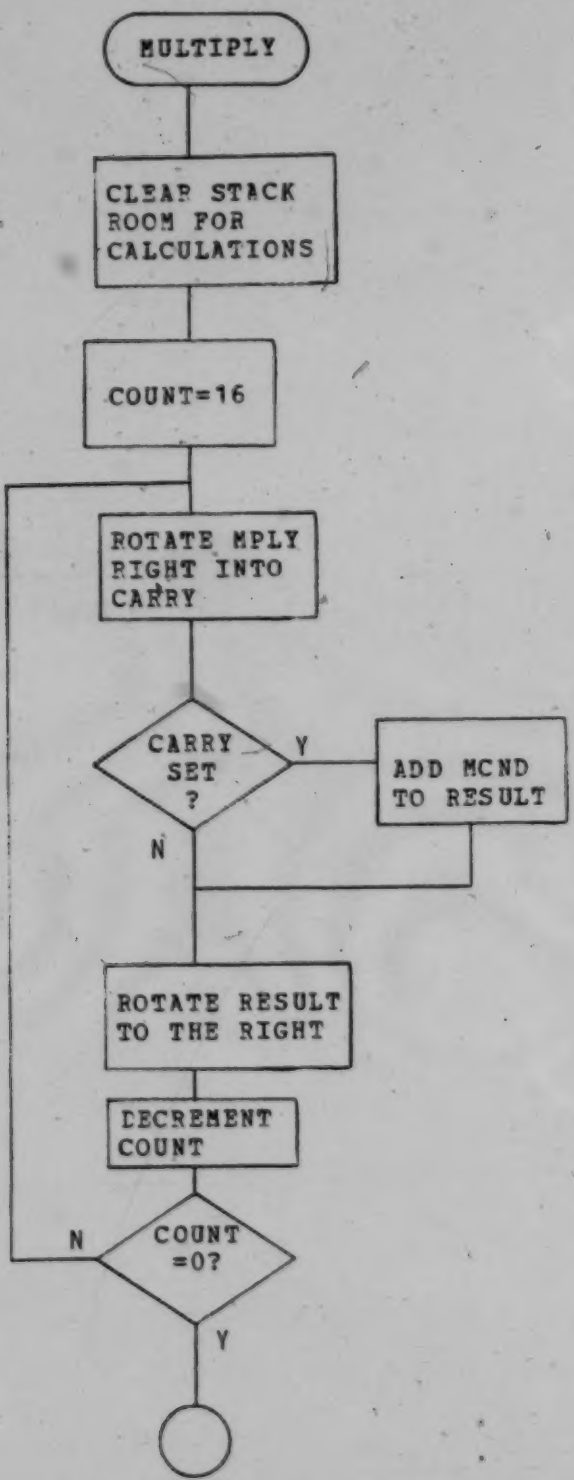
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAM	0	0	0	x	x	x										
KEYBOARD	0	0	1	0	0	0	0	0	0	0	0	0	0	x	x	
TIMER	0	0	1	0	0	0	0	0	0	0	0	0	1			
D/A PIA	0	1	0	1	1	x	x	x	x	x	x	x	0	0		
STATUS PIA	0	1	0	1	1	x	x	x	x	x	x	x	0	1		
A/D PIA	0	1	0	1	1	x	x	x	x	x	x	x	1	0		
ROM 0	1	1	x	x	0	0										
ROM 1	1	1	x	x	0	1										
ROM 2	1	1	x	x	1	0										
ROM 3	1	1	x	x	1	1										

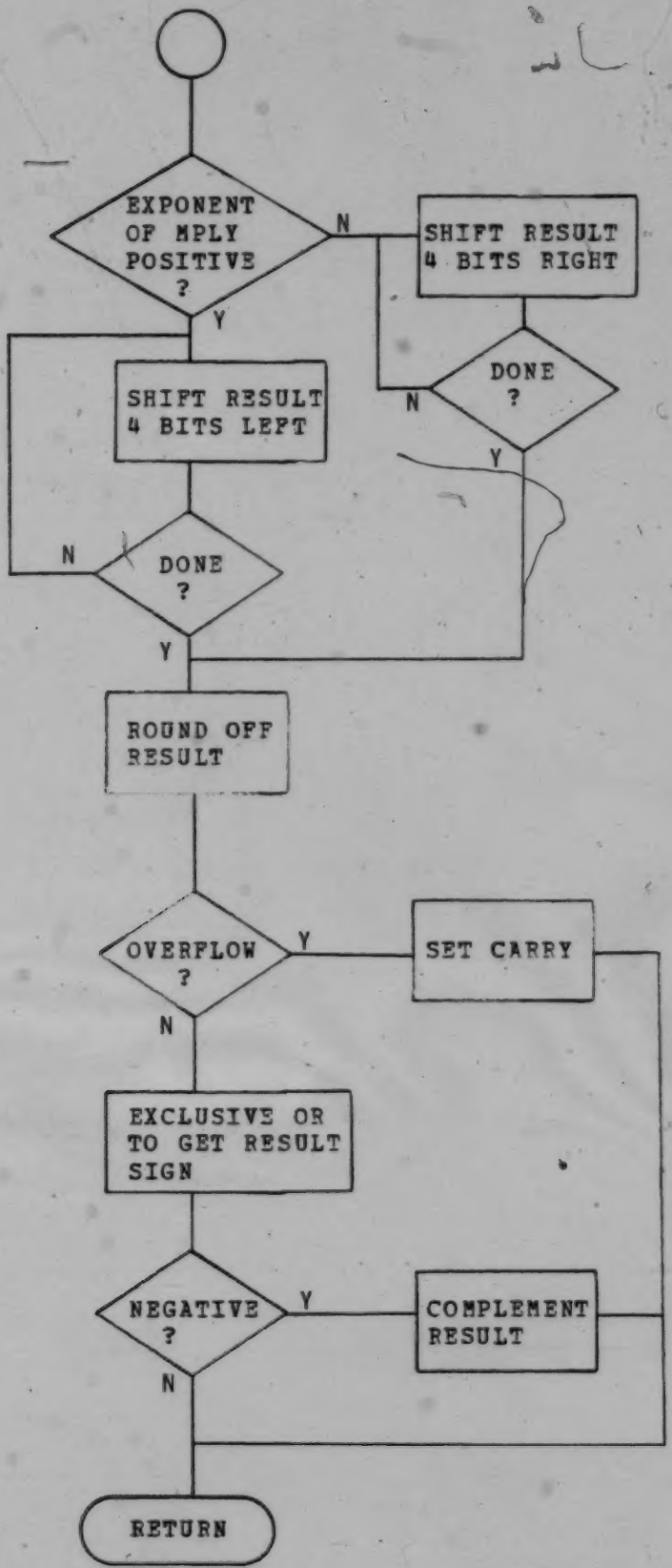
1. x is a don't care bit and is not decoded.
2. A blank address bit is used by the device at that address.
3. ROM 0 is not used and is a spare for future expansion.
4. ROM 1 contains the control algorithms.
5. ROM 2 contains the multiply routine, the restart routine, and other utility programs.
6. ROM 3 contains the keybpard, timer, and display control routines.
7. STATUS PIA is used to control the overload indicators.
8. KEYBOARD also controls the digital display.
9. RAM addresses 0000 to 00D5 (Hex) are used by the control program.
10. RAM addresses 03A0 to 03FF (Hex) are used by the stack.

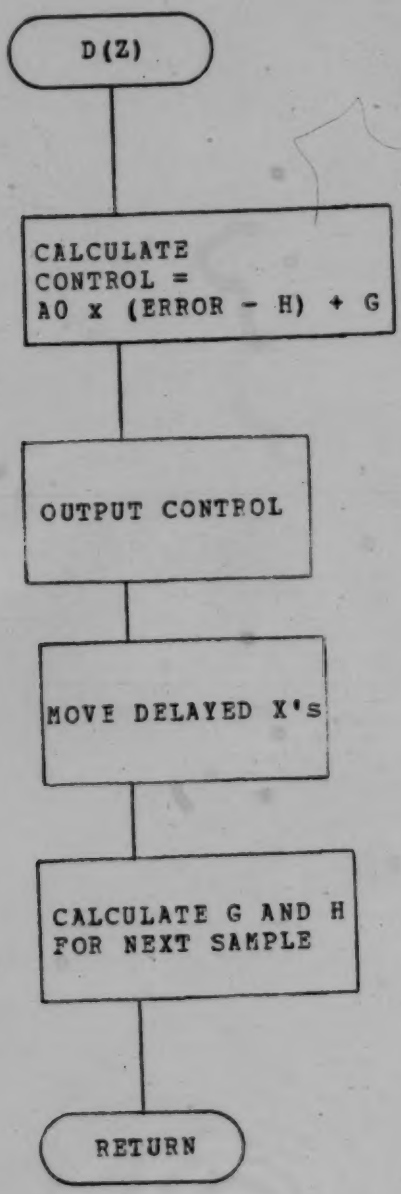
System Flowcharts

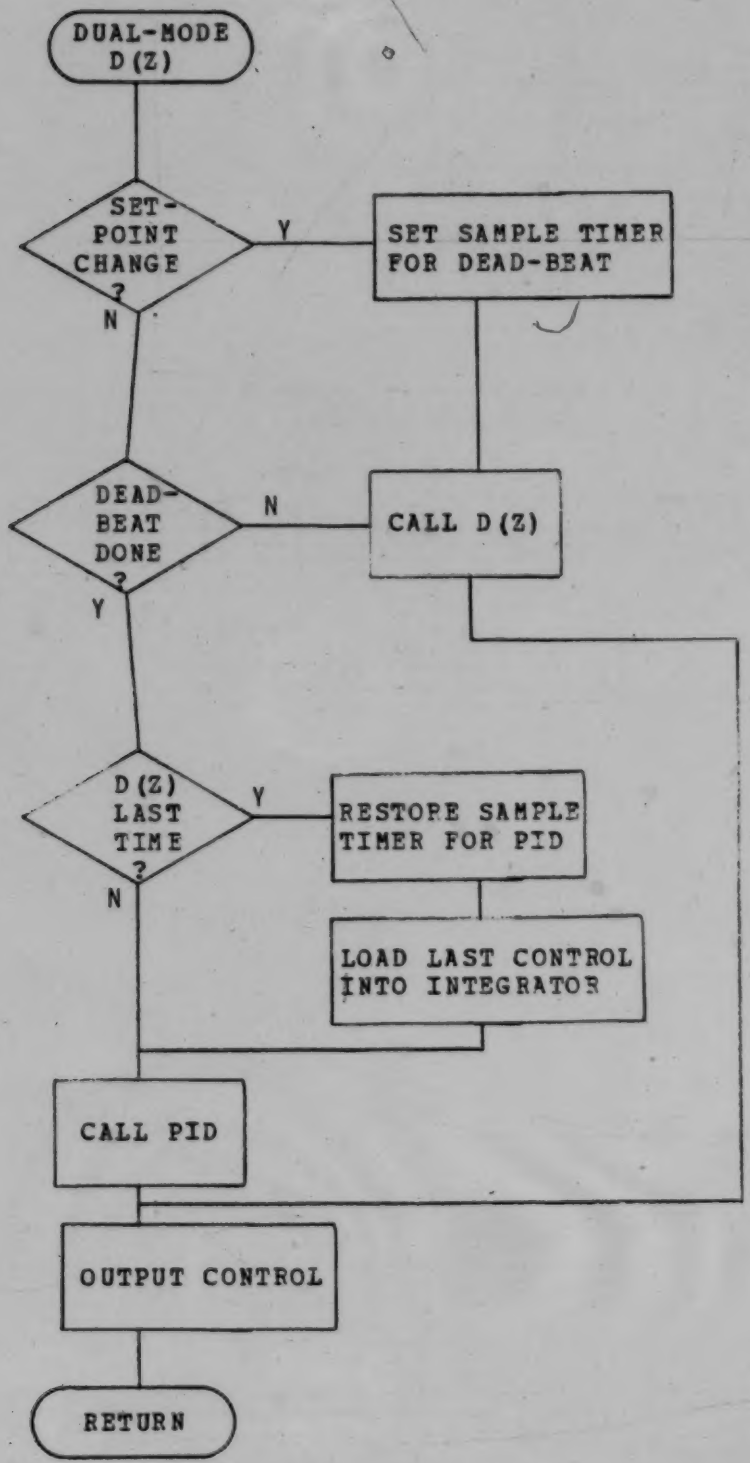


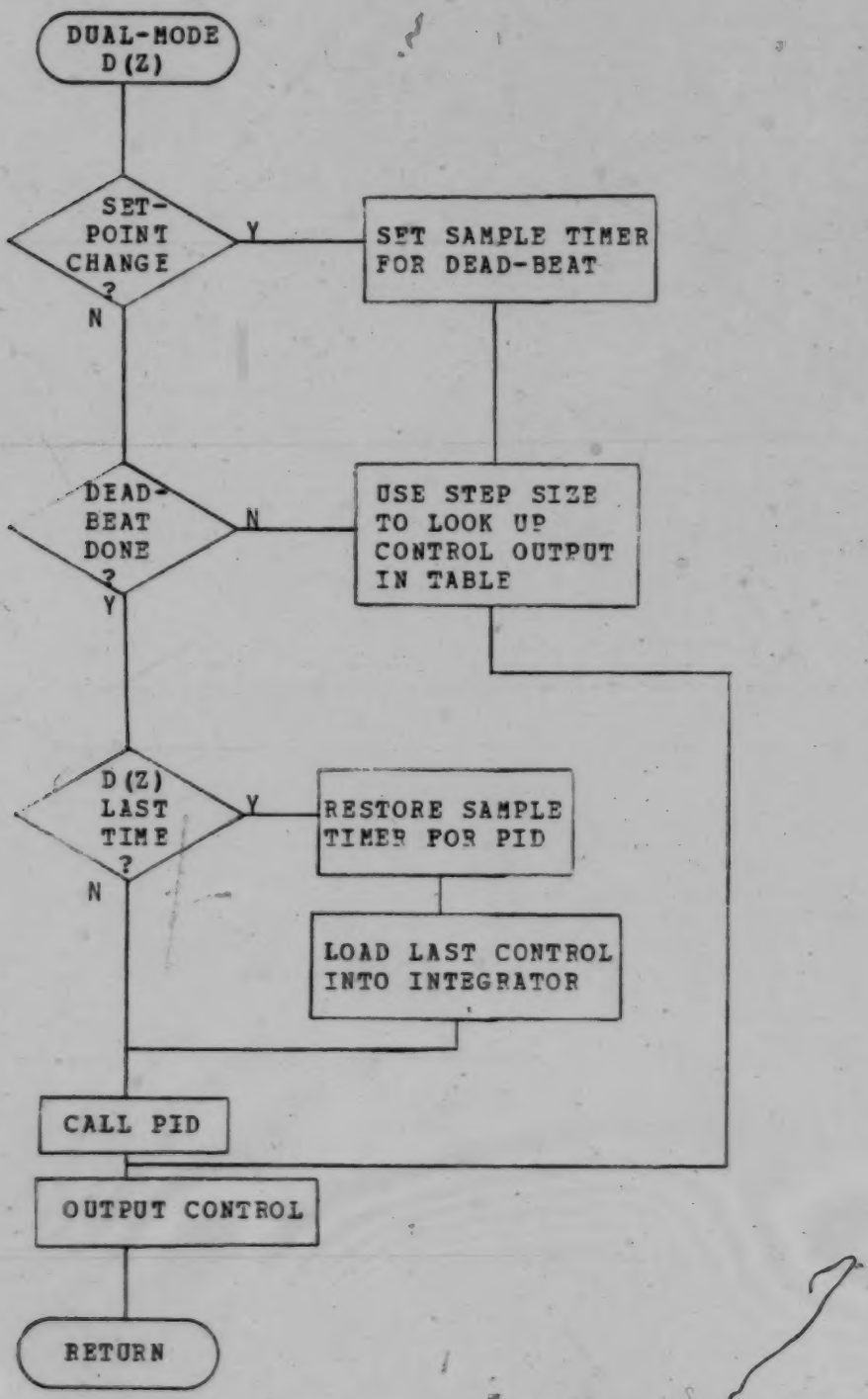

PID
CALCULATE
P=Kp x ERROR
CALCULATE
I=(Ki x ERROR) + I(k-1)
DIF(n) = ERROR(k-n) - ERROR(k-1-n)
D = Kd x (DIF(0) + DIF(1) + DIF(2) + DIF(3)) / 4
CALCULATE
CONTROL=P + I + D
OUTPUT CONTROL TO D/A
RETURN

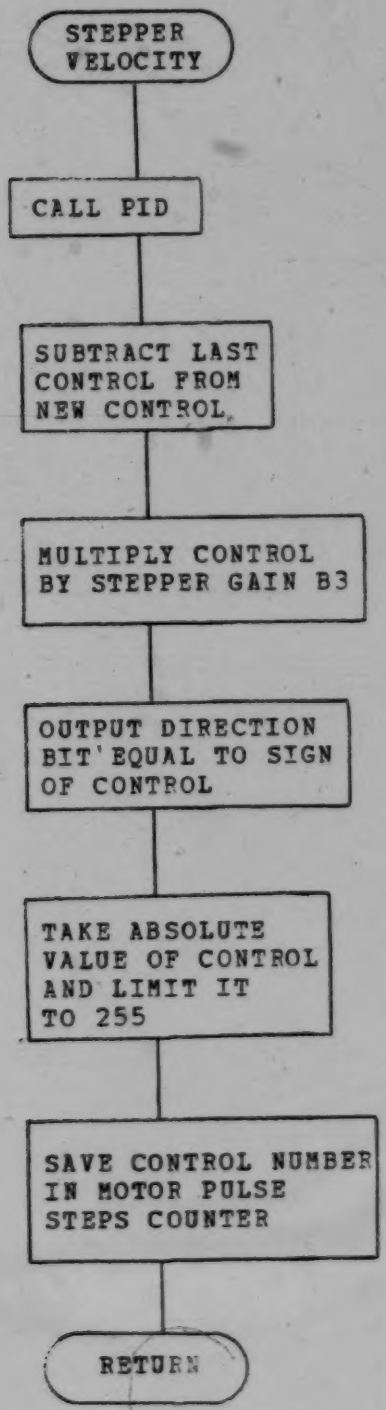

MULTIPLY
CLEAR STACK ROOM FOR CALCULATIONS
COUNT=16
ROTATE MPLY RIGHT INTO CARRY
CARRY SET ?
Y
ADD MCND TO RESULT
N
ROTATE RESULT TO THE RIGHT
DECREMENT COUNT
COUNT =0?
N
Y

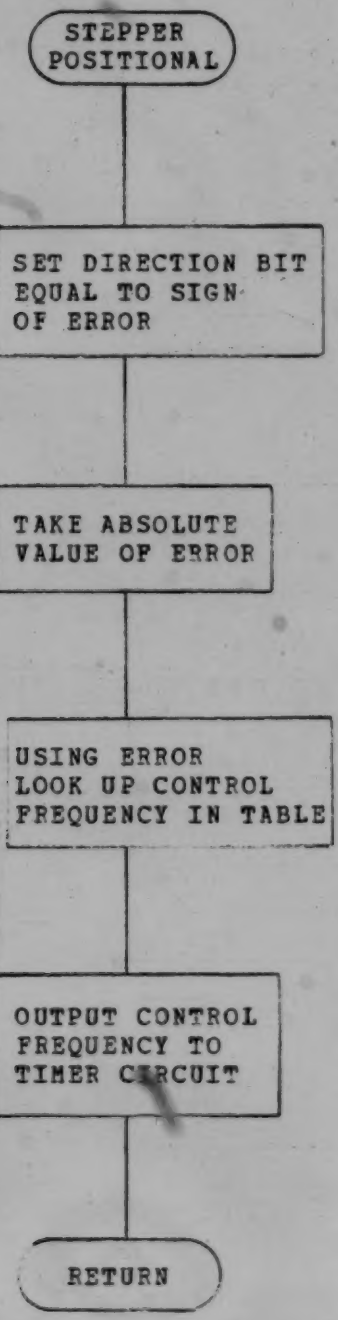

EXPONENT OF MPLY POSITIVE ?
N
SHIFT RESULT 4 BITS RIGHT
Y
SHIFT RESULT 4 BITS LEFT
DONE ?
N
Y
N
DONE ?
Y
ROUND OFF RESULT
OVERFLOW ?
Y
SET CARRY
N
EXCLUSIVE OR TO GET RESULT SIGN
NEGATIVE ?
Y
COMPLEMENT RESULT
N
RETURN

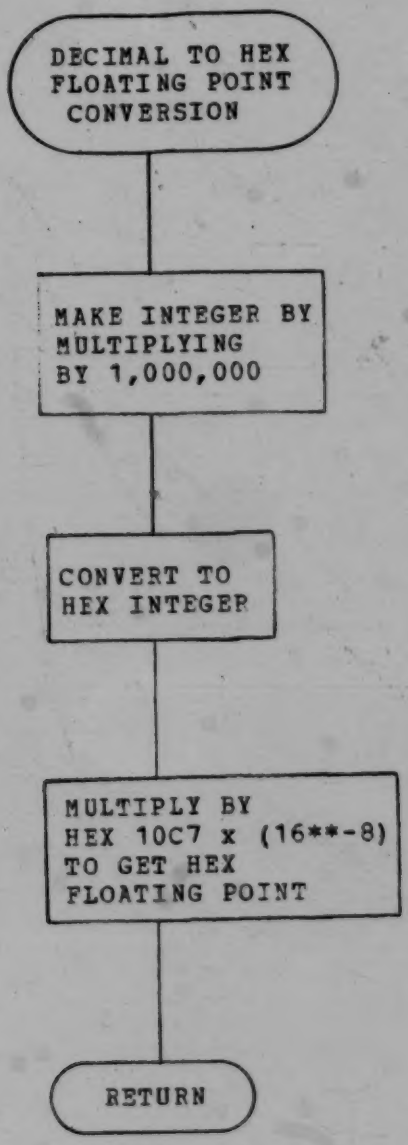

D(Z)
CALCULATE
CONTROL =
A0 x (ERROR - H) + G
OUTPUT CONTROL
MOVE DELAYED X's
CALCULATE G AND H
FOR NEXT SAMPLE
RETURN


DUAL-MODE D(Z)
SET-POINT CHANGE ?
Y
SET SAMPLE TIMER FOR DEAD-BEAT
N
DEAD-BEAT DONE ?
N
CALL D(Z)
Y
D(Z) LAST TIME ?
Y
RESTORE SAMPLE TIMER FOR PID
N
LOAD LAST CONTROL INTO INTEGRATOR
CALL PID
OUTPUT CONTROL
RETURN

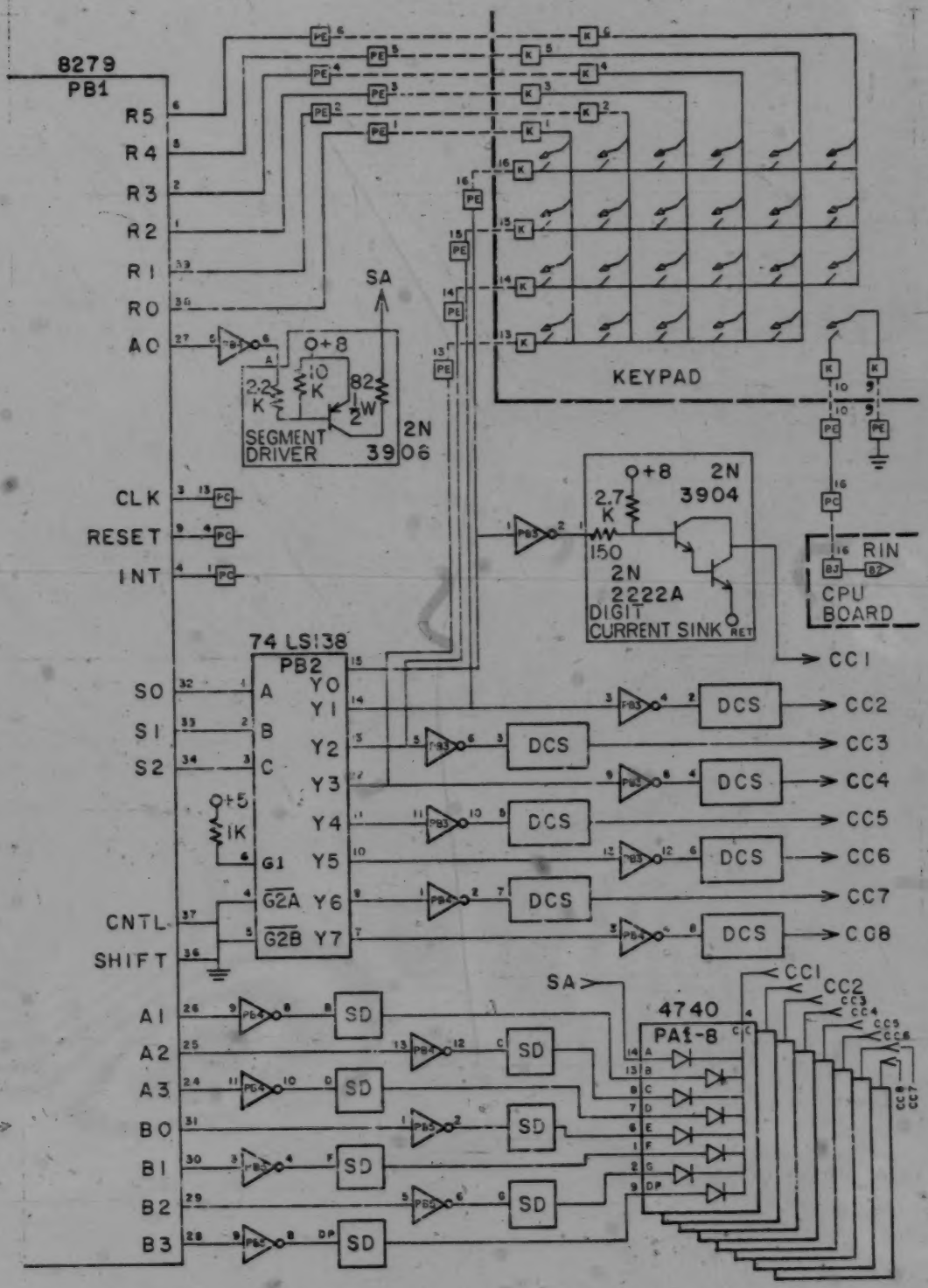

DUAL-MODE D(Z)
SET-POINT CHANGE ?
Y
N
SET SAMPLE TIMER FOR DEAD-BEAT
DEAD-BEAT DONE ?
N
Y
USE STEP SIZE TO LOOK UP CONTROL OUTPUT IN TABLE
D(Z) LAST TIME ?
Y
N
RESTORE SAMPLE TIMER FOR PID
LOAD LAST CONTROL INTO INTEGRATOR
CALL PID
OUTPUT CONTROL
RETURN

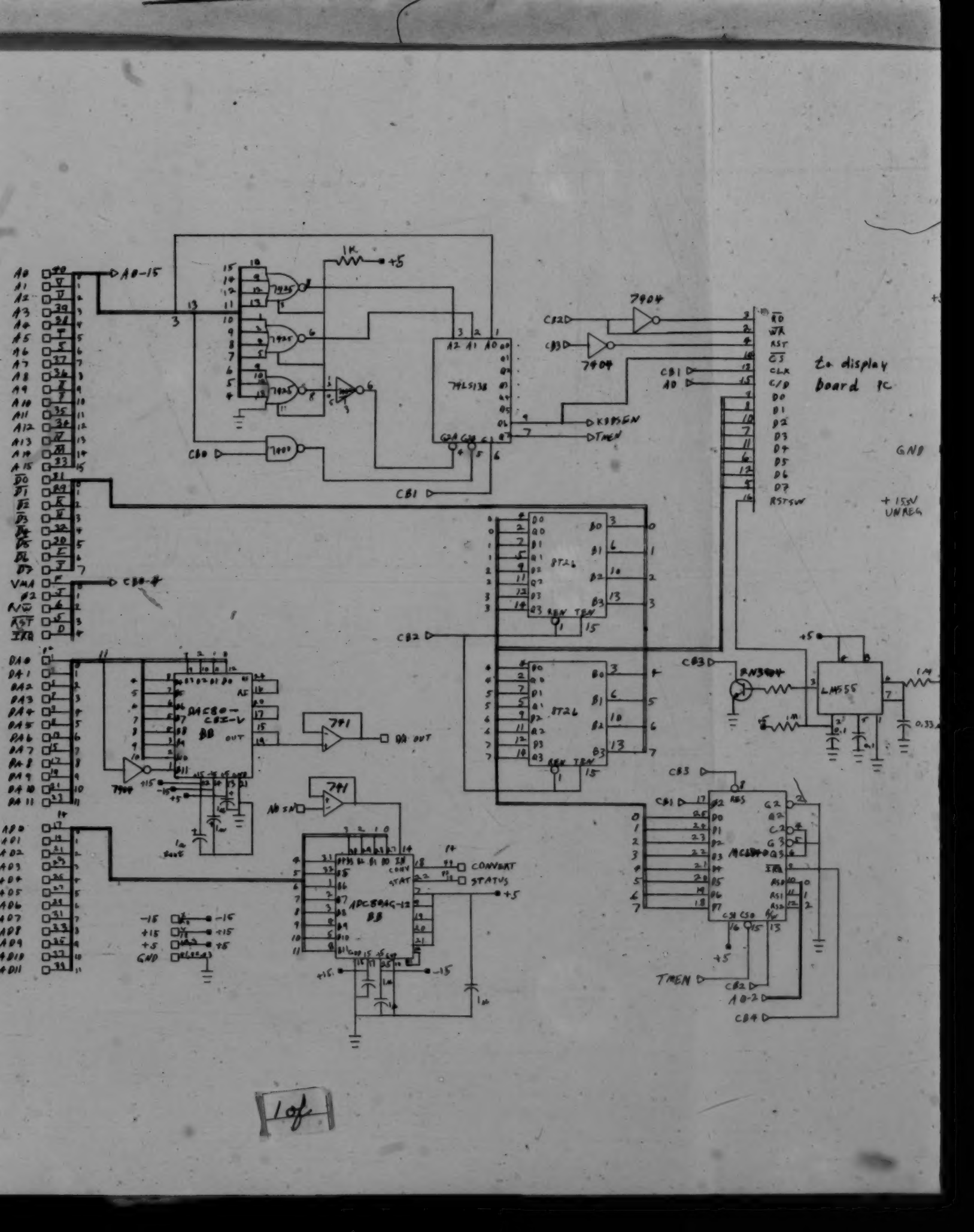

STEPPER VELOCITY
CALL PID
SUBTRACT LAST CONTROL FROM NEW CONTROL
MULTIPLY CONTROL BY STEPPER GAIN B3
OUTPUT DIRECTION BIT EQUAL TO SIGN OF CONTROL
TAKE ABSOLUTE VALUE OF CONTROL AND LIMIT IT TO 255
SAVE CONTROL NUMBER IN MOTOR PULSE STEPS COUNTER
RETURN

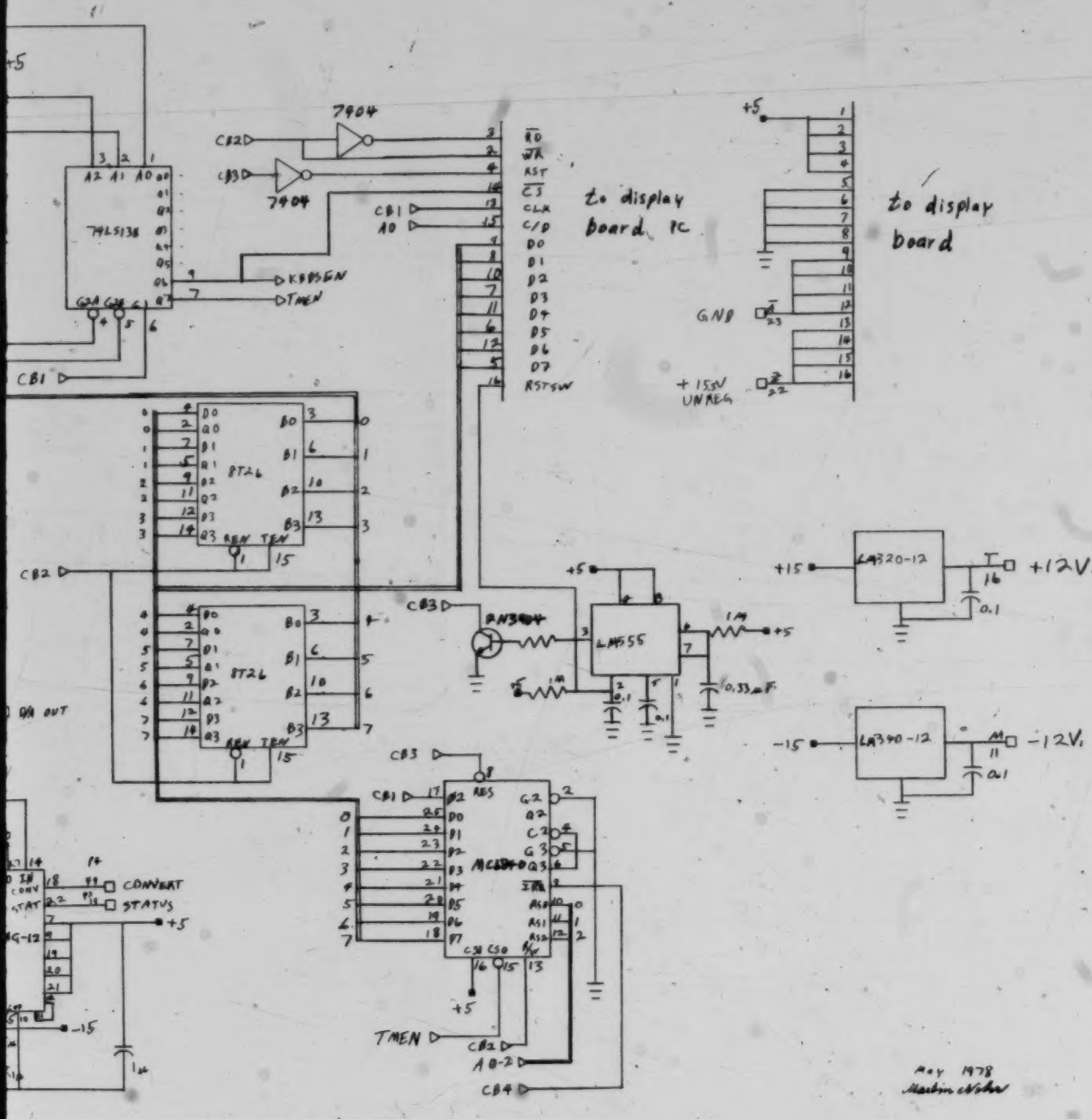

STEPPER POSITIONAL
SET DIRECTION BIT EQUAL TO SIGN OF ERROR
TAKE ABSOLUTE VALUE OF ERROR
USING ERROR LOOK UP CONTROL FREQUENCY IN TABLE
OUTPUT CONTROL FREQUENCY TO TIMER CIRCUIT
RETURN


DECIMAL TO HEX
FLOATING POINT
CONVERSION
MAKE INTEGER BY
MULTIPLYING
BY 1,000,000
CONVERT TO
HEX INTEGER
MULTIPLY BY
HEX 10C7 x (16**-8)
TO GET HEX
FLOATING POINT
RETURN

Schematic Diagrams


8279
PB1
R5
R4
R3
R2
R1
R0
A0
CLK
RESET
INT
CNTL
SHIFT
A1
A2
A3
B0
B1
B2
B3
SA
SEGMENT DRIVER
2N 3906
KEYPAD
2N 3904
2.7 K
150
2N 2222A
DIGIT CURRENT SINK
RET
RIN
CPU BOARD
CC1
74 LS138
PB2
SO
S1
S2
G1
G2A
G2B
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
DCS
CC2
CC3
CC4
CC5
CC6
CC7
CC8
SD
4740
PA1-8


A0-15
1K
+5
7425
74LS138
7404
CB2
CB3
CB1
CB0
A0
KBDSEN
TMEN
RD
WR
RST
CS
CLK
C/D
D0
D1
D2
D3
D4
D5
D6
D7
RSTSW
to display
Board pc
GND
+15V
UNREG
VMA
Ø2
R/W
RST
IRQ
CB0-4
8T24
CB2
CB3
2N3904
LM555
1M
0.33
DA0
DA1
DA2
DA3
DA4
DA5
DA6
DA7
DA8
DA9
DA10
DA11
DAC80-CBI-V
BB
OUT
7406
741
DA OUT
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
ADC804G-12
CONVERT
STATUS
MC6840
CB1
CB2
A0-2
CB4
TMEN
-15
+15
+5
GND
1 of 1


+5
7404
7404
CB2
CB3
CB1
A0
74LS138
A2 A1 A0
G2A G2B G1
KBDSEN
TMEN
CB1
RD
WR
RST
CS
CLR
C/D
D0
D1
D2
D3
D4
D5
D6
D7
RSTSW
to display board PC
to display board
+5
GND
+15V UNREG
8T26
8T26
REN TEN
CB2
CB3
PN3904
LM555
1M
0.33μF
0.1
+15
LM320-12
+12V.
-15
LM340-12
-12V.
0.1
CB3
CB1
RES
CS1 CS0
+5
TMEN
CB2
A0-2
CB4
CONVERT
STATUS
+5
-15
1μ
May 1978
Martin Usher
2 of 2

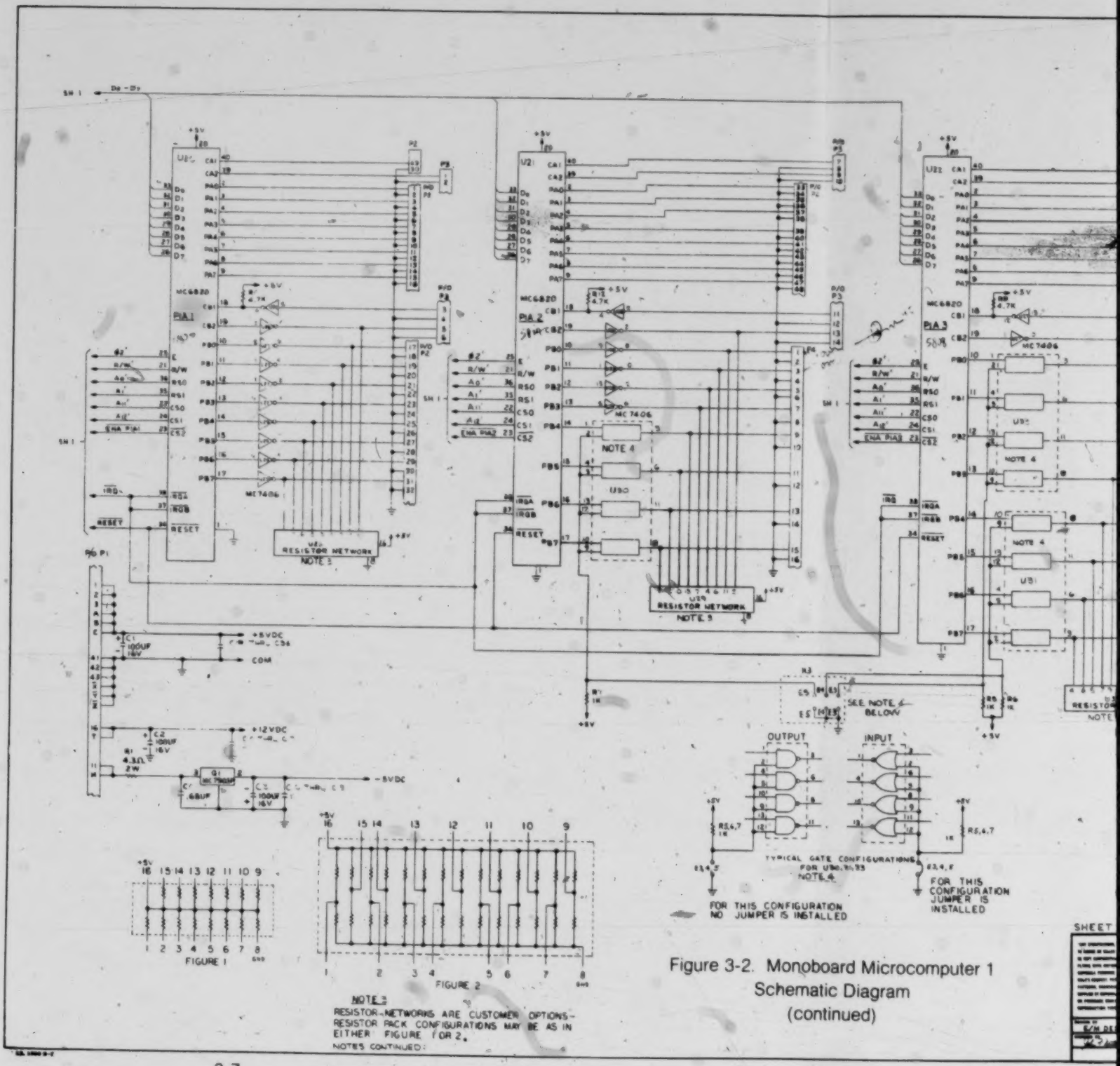


Figure 3-2. Monoboard Microcomputer 1 Schematic Diagram (continued)

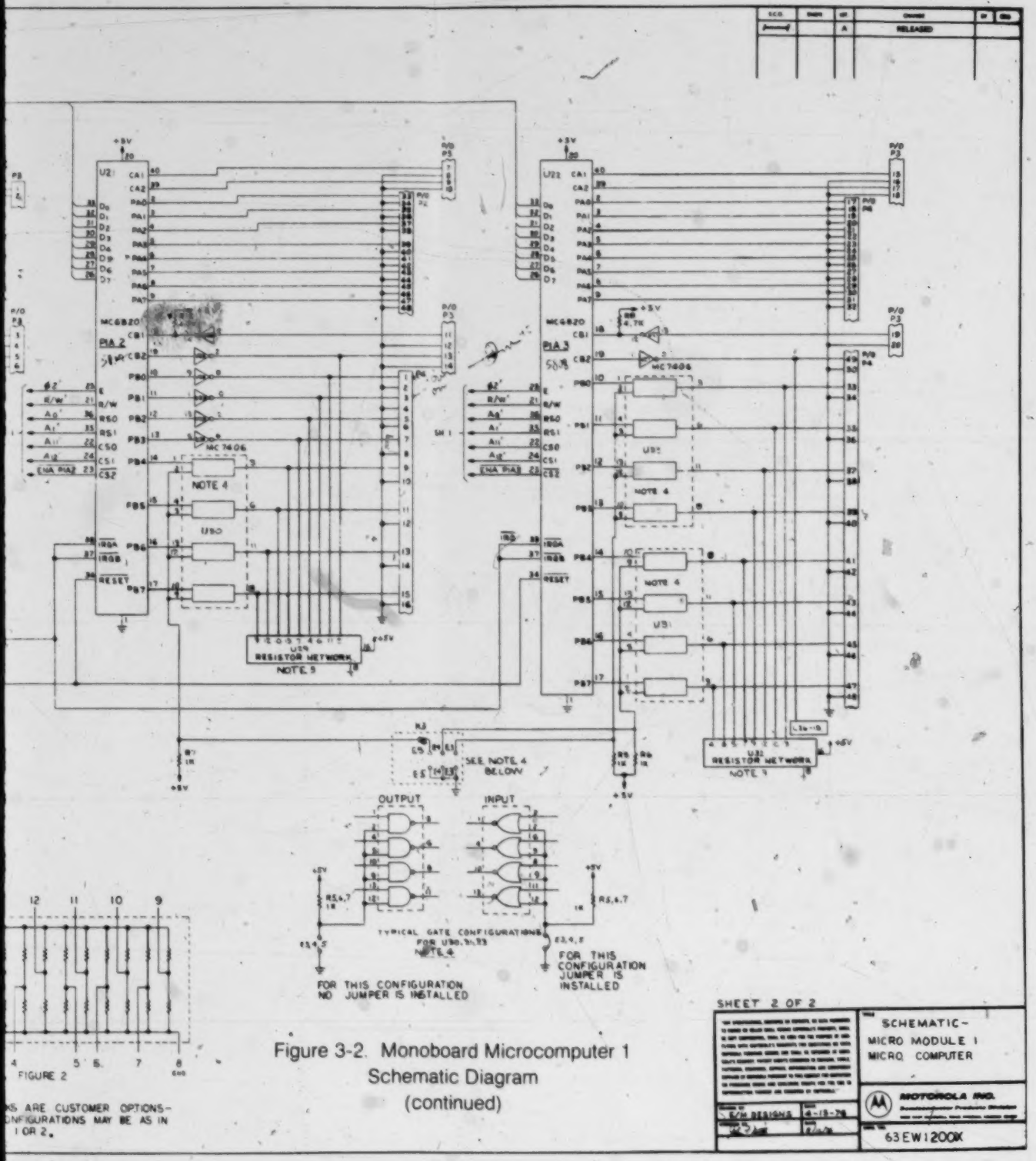


Figure 3-2. Monoboard Microcomputer 1 Schematic Diagram (continued)

2 of 2

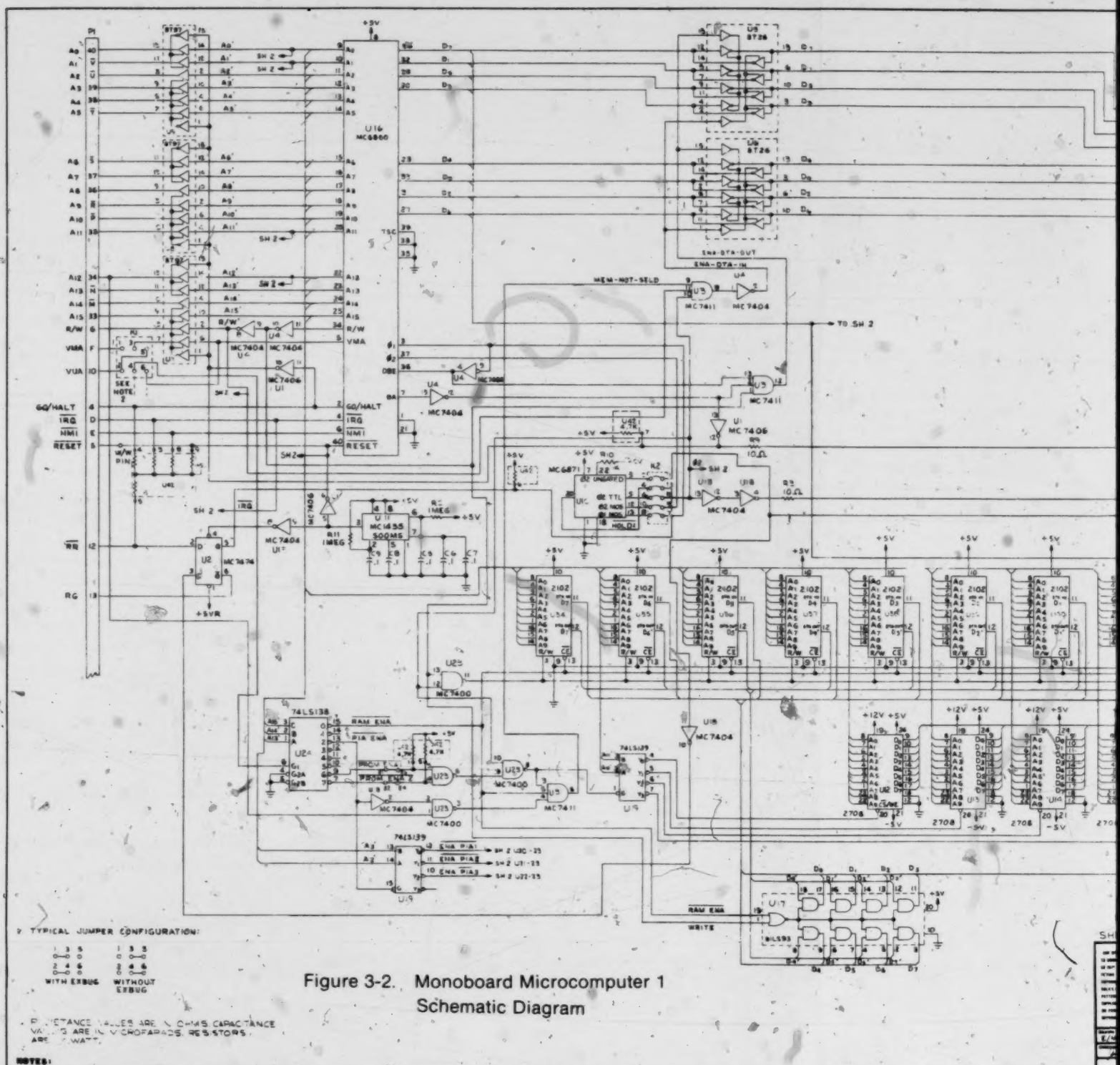
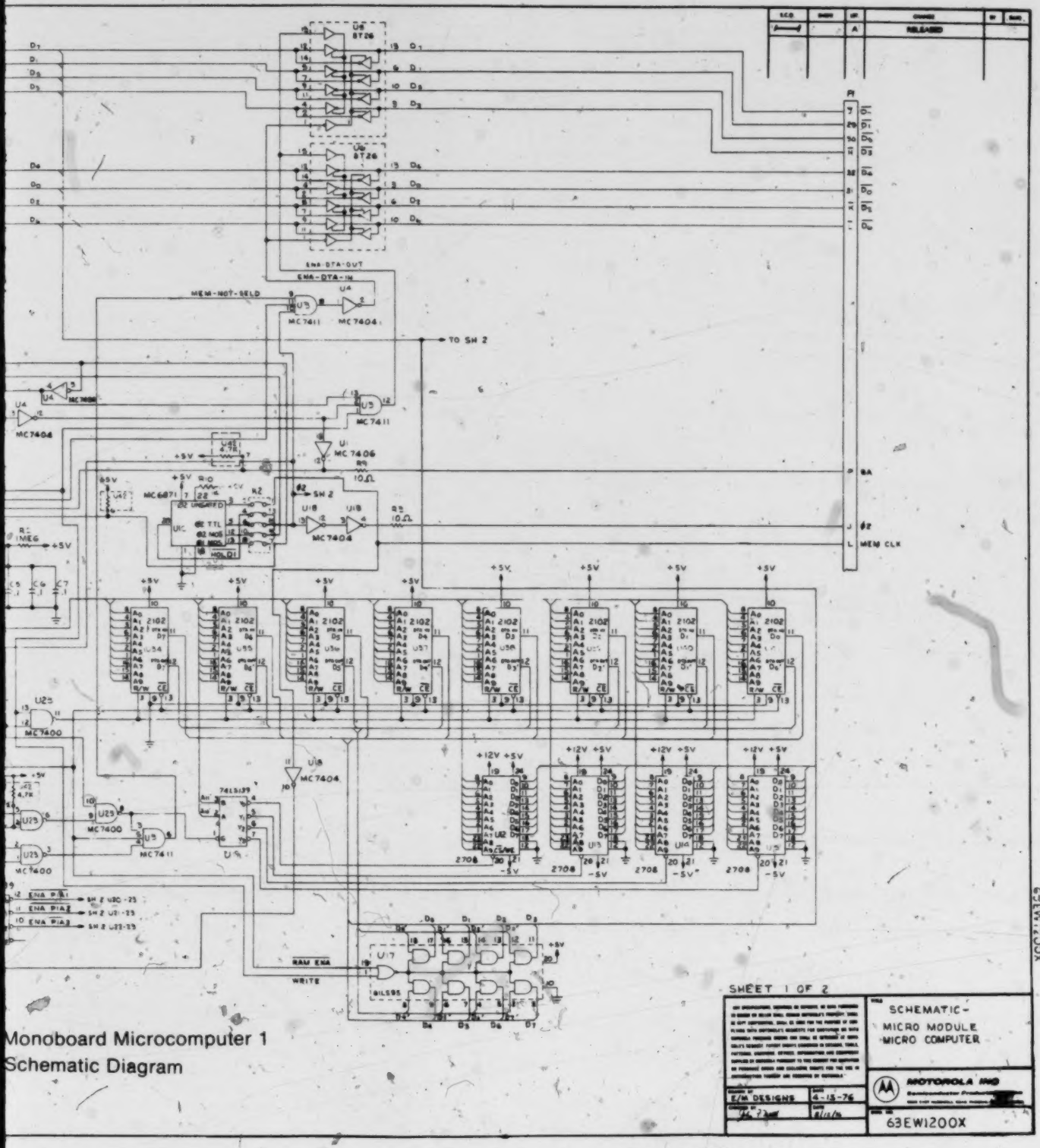


Figure 3-2. Monoboard Microcomputer 1
Schematic Diagram



Monoboard Microcomputer 1
Schematic Diagram

2 of 2

The Program Source Code

```
 NAM ROM
 ORG $0000
DCTR RMB 1 ;DIGIT DISPLAY POSITION COUNTER
KP RMB 3 ;PROPORTIONAL GAIN CONSTANT
LSMPL RMB 2 ;PREVIOUS SAMPLE VALUE
SMPL RMB 2 ;PRESENT SAMPLE VALUE STORAGE
TKP RMB 3 ;TEMPORARY STORAGE FOR KP
SPFN RMB 2 ;FUNCTION IN USE FLAG
KI RMB 3 ;INTEGRAL GAIN CONSTANT
KD RMB 3 ;DERIVATIVE GAIN CONSTANT
XTEMP RMB 2 ;TEMPORARY STORAGE FOR X
CNTL RMB 2 ;CONTROL VALUE STORAGE
ISMPL RMB 2 ;INTEGRAL CONTROL
DSMPL RMB 2 ;DERIVATIVE CONTROL
TMP RMB 3 ;TEMPORARY STORAGE FOR COEFFICIENTS
*
KPT RMB 6 ;STORAGE OF BCD VALUES
KIT RMB 6 ; USED FOR THE DISPLAY SO THAT HEX
KDT RMB 6 ; NEED NOT BE TRANSLATED TO DECIMAL
*
KIND RMB 6 ;INPUT TO DECIMAL TO HEX
KINI RMB 9 ;STORAGE LOCATIONS USED BY
KINB RMB 4 ; THE ROUTINE TO TRANSLATE
KINBT RMB 4 ; DECIMAL INTO HEX
KINHF RMB 3 ;THIS HAS THE RESULT IN IT
*
SMPT RMB 6 ;SAMPLE TIME DISPLAY STORAGE
SMPTT RMB 6 ;TEMPORAY STORAGE FOR DISPLAY OF TIME
DS RMB 8
CNTMD RMB 1 ;CONTRCL MODE
*
A0 RMB 3 ;COEFFS FOR D(Z)
A1 RMB 3
A2 RMB 3
A3 RMB 3
B1 RMB 3
B2 RMB 3
B3 RMB 3
A0T RMB 6 ;DISPLAY DIGITS STORAGE
A1T RMB 6
A2T RMB 6
A3T RMB 6
B1T RMB 6
B2T RMB 6
B3T RMB 6
G RMB 2 ;CALCULATION OF G
H RMB 2 ;CALCULATION OF H
X0 RMB 2 ;DELAYED X'S FOR D(Z)
X1 RMB 2
X2 RMB 2
X3 RMB 2
DZN RMB 1 ;ORDER OF D(Z)
```

```
DZNT RMB 1 ;FOR TEMP USE OF D(Z)
*
DCML RMB 1 ;USED TO SET DECIMAL
SETPNT RMB 2 ;THIS IS THE SETPOINT
BMPFLG RMB 1 ;BUMPLESS TRANSFER FLAG
SETPD RMB 6 ;DISPLAY OF SET POINT
DCSP RMB 2 ;DECIMAL SET POINT
TSRC RMB 2 ;SOURCE SAVE FOR MOVE
TDST RMB 2 ;DESTINATION FOR MOVE
PPX RMB 3 ;USED BY PUSH AND PULL X
DSPLY RMB 6 ;TEMP SAVE FOR DISPLAY
DZPFLG RMB 2 ;DUAL MODE CONTROLLER FLAG
TMSV RMB 4 ;TIMER VALUES
STEPS RMB 1 ;STEPPER MOTOR COUNTER
LCNTL RMB 2 ;LAST CONTROL OUTPUT
TSTP RMB 2 ;STEP PULSE TIME STORAGE
STTMD RMB 6 ;PULSE TIME DISPLAY
MOVE EQU $C800 ;MOVE ROUTINE ENTRY POINT
CNTLP EQU $C400 ;CONTROL ENTRY POINT
WDAI EQU $90 ;WRITE DISPLAY AUTOINC
WD EQU $80 ;WRITE DISPLAY
RB EQU $40 ;READ BUTTON
RDAI EQU $70 ;READ DISPLAY
CD EQU $D0 ;CLEAR DISPLAY
KDSP EQU $2000 ;BUTTON AND DISPLAY ADDRESS
FMLT EQU $C80C ;MULTIPLY PROGRAM ENTRY POINT
DA EQU $5800 ;D/A PIA ADDRESS
AD EQU $5808 ;A/D PIA ADDRESS
TIMER EQU $2008 ;TIMER ADDRESS
CVLD EQU $5804 ;OVERLOAD PIA ADDRESS
TCR3 EQU $82 ;TIMER 3 CONTROL REGISTER
TCR2 EQU $81 ;TIMER 2 CONTROL REGISTER
TSTC EQU $24 ;CONTROL FOR STEPPER TIMER
TCR1 EQU $10 ;TIMER 1 CONTROL REGISTER
TOEN EQU $80 ;TIMER 1 OUTPUT ENABLE
TICR EQU $40 ;INTERRUPT CONTROL FOR TIMERS
STRT EQU $C809 ;START-UP ROUTINE
GHDIN EQU $C803 ;DIGITS INPUT TO COEFFICIENTS
*
* THIS IS THE LOWEST ADDRESS ROM
 ORG $C400
*
* THIS ROUTINE WILL PERFORM
* THE DIFFERENT CONTROL
* OPERATIONS
CNTLP LDA A TIMER+1 ;GET STATUS
 BIT A #$02 ;IS IT SAMPLE TIME?
 BNE CNW
 BIT A #$01 ;IS IT STEPPER?
 BEQ CNTX ;NO
 TST STEPS
 BEQ CNT1 ;ALREADY ZERO
 DEC STEPS ;OTHERWISE DECREMENT IT
 LDX TSTP ;PULSE OUT TO MOTOR
```

```
 STX TIMER+2
 BRA CNT2
CNT1 LDA A #TSTC ;TURN OFF INTERRUPTS AND OUTPUT
 STA A TIMER
CNT2 LDA A TIMER+1
 LDX TIMER+2 ;CLEAR INTERRUPT
 BRA CNTX ;EXIT
CNW CLR AD+2 ;START SAMPLE
* FIND CONTROL MODE
* AND JUMP TO IT AFTER GETTING SAMPLE
 LDX #CNTBL
 LDA B CNTMD
 JSR ULDE
CNW1 TST AD+3 ;WAIT FOR SAMPLE
 BPL CNW1
 LDA A AD+2 ;GET MSB
 LDA B AD ;GET LSB
 COM B ;COMPLEMENT IT
 AND A #$0F
 BIT A #$08 ;TEST FOR NEGATIVE
 BEQ SPS
 ORA A #$F0
SPS STA A SMPL
 STA B SMPL+1
* CALCULATE SET POINT
* MINUS THE SAMPLE
 LDA A SETPNT ;GET SET POINT
 LDA B SETPNT+1
 SUB B SMPL+1 ;SUBTRACT SAMPLE
 SBC A SMPL
 STA A SMPL ;SAVE IN SAMPLE
 STA B SMPL+1
*
* NOW JUMP TO CONTROL MODE
 JSR 0,X
 LDX SMPL
 STX LSMPL ;SAVE LAST SAMPLE
 LDX CNTL ;SAVE LAST CONTROL
 STX LCNTL
 LDA A TIMER+1
 LDX TIMER+4 ;CLEAR TIMER 2 INTERRUPT
 LDA A TIMER+1 ;CLEAR TIMER 1 INTERRUPT
 LDX TIMER+2
CNTX RTI
* THIS IS THE PID CONTROLLER
PIDC BSR PIDCC
 JSR SMOUT
 JSR PIDCD
 RTS
*
PIDCC JSR FMLT ;MULTIPLY BY
 FDB SMPL ;KP
 FDB KP
 STA A CNTL
```

```
 STA B CNTL+1
 TST KI+1
 BNE IC
 CLR ISMPL
 CLR ISMPL+1
 BRA DC
IC JSR FMLT ;CALCULATE INTEGRAL
 FDB SMPL
 FDB KI
 LDX #ISMPL
 JSR DADD
 LDX #CNTL ;ADD TO OUTPUT
 JSR DADD
* SKIP DERIVATIVE WHILE BUMPLESS IS SET
DC TST BMPFLG
 BEQ DCA
 DEC BMPFLG
 CLR DS
 CLR DS+1
 RTS
DCA LDA A SMPL ;CALCULATE DERIVATIVE
 LDA B SMPL+1
 SUB B LSMPL+1
 SBC A LSMPL
 STA A DS
 STA B DS+1
* TAKE AVERAGE OF LAST 4 SLOPES
* A,B HAS PRESENT SLOPE
 ADD B DS+3
 ADC A DS+2
 ADD B DS+5
 ADC A DS+4
 ADD B DS+7
 ADC A DS+6
 ASR A ;DIVIDE BY FOUR
 ROR B
 ASR A
 ROR B
 STA A DSMPL
 STA B DSMPL+1
 JSR FMLT
 FDB DSMPL
 FDB KD
 LDX #CNTL
 JSR DADD
 RTS
* NOW MOVE DELAYED DERIVATIVES
FIDCD LDX DS+4
 STX DS+6
 LDX DS+2
 STX DS+4
 LDX DS
 STX DS+2
 RTS
```

```
* REDUCE TO 12 BITS AND OUTPUT CONTROL WORD
SMOUT PSH A ;SAVE MSB
 AND A #$F8 ;CHECK FOR
 BEQ NSOV ;12 BIT
 CMP A #$F8 ;OVERFLOW
 BEQ NSOV
 PUL A ;OVERFLOW
 TST A ;SET TO
 BMI NGOV ;MAX
PSOV LDA A #1
 STA A OVLD+2
 LDA A #$07
 LDA B #$FF
 BRA COUTP
NGOV LDA A #2
 STA A OVLD+2
 LDA A #$08
 CLR B
 BRA COUTP
NSOV CLR OVLD+2
 PUL A
COUTP COM B
 STA A DA+2 ;OUTPUT IT
 STA B DA
 RTS
* D(Z) CONTROL MODE
DZC BSR DZCC
 JSR SMOUT ;OUTPUT IT
 BSR DZCD ;COMPUTE NEW G AND H
 RTS
DZCC SUB B H+1 ;CALCULATE NEW X
 SBC A H
 STA A X0
 STA B X0+1
* CALCULATE OUTPUT
 JSR FMLT
 FDB X0
 FDB A0
 LDX #G
 JSR DADD
 RTS
* SHIFT DELAYED X'S
DZCD LDX X2
 STX X3
 LDX X1
 STX X2
 LDX X0
 STX X1
* CALCULATE G FOR NEXT TIME
 LDA A DZN
 STA A DZNT
 CLR G
 CLR G+1
 JSR FMLT
```

```
 FDB X1
 FDB A1
 LDX #G
 JSR DADD
 DEC DZNT
 BEQ CALH
 JSR FMLT
 FDB X2
 FDB A2
 LDX #G
 JSR DADD
 DEC DZNT
 BEQ CALH
 JSR FMLT
 FDB X3
 FDB A3
 LDX #G
 JSR DADD
CALH LDA A DZN
 STA A DZNT
 CLR H
 CLR H+1
 JSR FMLT
 FDB X1
 FDB B1
 LDX #H
 JSR DADD
 DEC DZNT
 BEQ GHX
 JSR FMLT
 FDB X2
 FDB B2
 LDX #H
 JSR DADD
 DEC DZNT
 BEQ GHX
 JSR FMLT
 FDB X3
 FDB B3
 LDX #H
 JSR DADD
GHX RTS
*
* DUAL MODE CONTROLLER
* SET POINT CHANGE CAUSES
* DEADBEAT CONTROL FOR
* N+2 SAMPLE PERIODS
* AT SAMPLE TIME ONE
* AND THEN RETURNS TO
* PID AT SAMPLE TIME ZERO
DZPC TST DZPFLG
 BEQ DZPC1
 TST DZPFLG+1 ;IS THIS D(Z) START?
 BNE DZPC2 ;NO
```

```
 INC DZPFLG+1 ;YES, SET FLAG
 LDX TMSV+2 ;CHANGE TIMER
 STX TIMER+4 ;TO TIME ONE
DZPC2 DEC DZPFLG
 JSR DZC ;RUN D(Z) CONTROL
 JSR PLEI ;UPDATE PID INTEGRATOR
 RTS
DZPC1 TST DZPFLG+1
 BEQ DZPC4
 CLR DZPFLG+1 ;CLEAR FLAG
 LDX TMSV ;AND RESTORE TIMER
 STX TIMER+4 ;TO TIME ZERO
DZPC4 JSR PIDC
 RTS
*
* DOUBLE ADD
* RESULT IN A,B AND
* 0,X AND 1,X
* A,B ADDED TO 0,X , 1,X
* CARRY IS SET ON OVERFLOW
* AND RESULT IS SET TO MAX
*
DADD ADD B 1,X ;ADD LSB
 ADC A 0,X ;ADD MSB
 BVC DADD1
 BLT DADD2
 LDA A #$7F
 LDA B #$FF
 BRA DADD3
DADD2 LDA A #$80
 CLR B
DADD3 SEC
 BRA DADD4
DADD1 CLC
DADD4 STA A 0,X
 STA B 1,X
 RTS
*
* TABLE FOR CONTROL MODES
CNTBL FDB PIDC,DZC,DZPC,TDBT,VALG,STM
*
* DUAL MODE CONTROLLER
* USING TABLE LOOK UP FOR
* OUTPUT VALUES
TDBMT FDB AM0,AM1,AM2,AM3 ;TABLE FOR OUTPUTS
TDBT TST DZPFLG ;PID AGAIN
 BEQ TDBT1 ;WAS LAST TIME DEAD BEAT?
 TST DZPFLG+1 ;IS THIS DEAD BEAT START?
 BNE TDBT3
 LDX TMSV+2 ;SWAP TIMER
 STX TIMER+4 ;TO TIME ONE
 INC DZPFLG+1 ;SET FLAG
TDBT3 LDA B DZN
 SUB B DZPFLG
```

```
INC B ;B HAS NUMBER OF OUTPUT
LDX #TDBMT
JSR ULDE ;X POINT TO SUBROUTINE
JSR 0,X ;CALCULATE OUTPUT VALUE
JSR SMOUT ;OUTPUT IT
DEC DZPFLG
JSR PLEI ;UPDATE PID INTEGRATOR
BRA TDBT4
TDBT1 TST DZPFLG+1
BEQ TDBT2 ;GOTO PID
CLR DZPFLG+1
LDX TMSV ;RESTORE TIMER ZERO
STX TIMER+4
TDBT2 JSR PIDC
TDBT4 RTS
AM0 JSR FMLT
FDB PPX
FDB A0
RTS
AM1 JSR FMLT
FDB PPX
FDB A1
RTS
AM2 JSR FMLT
FDB PPX
FDB A2
RTS
AM3 JSR FMLT
FDB PPX
FDB A3
RTS
* PUT LAST OUTPUT INTO THE PID INTEGRATOR
PLEI LDA A DA+2
LDA B DA
COM B
STA A ISMPL
STA B ISMPL+1
RTS
*
* THIS IS THE POSITIONAL STEPPER MOTOR CONTROL PART
STM TST A
BNE ST2
CMP B #$14
BLS ST3
BRA ST1
ST2 CMP A #$FF
BNE ST1
CMP B #$EC
BLS ST1
ST3 PSH A
LDA A #TCR1
STA A TIMER
PUL A
JSR PIDCC
```

```
 JSR PIDCD
 RTS
ST1 PSH A
 LDA A #TCR1  TOEN ;ENABLE THE TIMER OUTPUT
 STA A TIMER
 PUL A
 JSR PIDCC ;GET THE PID CONTROL
 PSH A ;SAVE MSB
 PSH A ;WE NEED IT TWICE
 AND A #$F8 ;CHECK FOR 12 BIT OVERFLOW
 BEQ STM0 ;GOOD
 CMP A #$F8 ;OVERFLOW IF FIRST 5 BITS NOT SAME
 BEQ STM0 ;GOOD
 PUL A ;GET A BACK
 TST A ;SET TO
 BMI NSTM ;MAX IF OVERFLOWED
PSTM LDA A #$07 ;POSITIVE OVERFLOW
 LDA B #$FF
 BRA STM00
NSTM LDA A #$F8 ;NEGATIVE OVERFLOW
 CLR B
 BRA STM00
STM0 PUL A
STM00 TST A
 BMI STM1 ;CHECK IF IT IS NEGATIVE
 LDA A #1
 STA A OVLD ;PLUS DIRECTION FOR STEPPER
 PUL A ;RESTORE A
 BRA STM2
STM1 CLR A
 STA A OVLD ;REVERSE DIRECTION FOR STEPPER
 PUL A ;RESTORE A
 COM A
 COM B
 ADD B #1 ;WE NEED ABSOLUTE VALUE SO,
 ADC A #0 ;COMPLEMENT A,B
STM2 LDX #SMTBL
STM3 PSH A ;SAVE A
* STOP WHEN THE NEXT TABLE ENTRY IS LARGER THAN A,B
 CMP B 5,X ;SUB M(X) FROM A,B
 SBC A 4,X
 PUL A ;GET A BACK
 BCS STM5 ;M(X) IS BIGGER THAN A,B
 INX
 INX
 INX
 INX ;TRY THE NEXT ONE
 BRA STM3
STM5 LDX 2,X ;GET THE TIMEOUT VALUE
 STX TIMER+2 ;OUTPUT NEW TIMER VALUE
 JSR PIDCD ;CLEAN UP AFTER THE PID CONTROL
 RTS
*
* THIS IS THE TABLE FOR THE STEPPER TIMER
```

```
* THE FIRST TWO BYTES ARE THE VOLTAGE RANGE
* AND THE NEXT TWO BYTES ARE THE CORRESPONDING
* STEP TIME, ETC.
SMTBL FDB $0000,$FFFF
 FDB $000A,$01F3
 FDB $000F,$014D
 FDB $0014,$00F9
 FDB $0029,$007C
 FDB $003D,$0052
 FDB $0052,$003E
 FDB $0066,$0031
 FDB $007B,$0029
 FDB $008F,$0023
 FDB $00A4,$001E
 FDB $00B8,$001B
 FDB $00CD,$0018
 FDB $00E1,$0016
 FDB $00F6,$0014
 FDB $010A,$0012
 FDB $011F,$0011
 FDB $0133,$0010
 FDB $019A,$000B
 FDB $0200,$0009
 FDB $0266,$0007
 FDB $02CD,$0006
 FDB $0333,$0005
 FDB $0400,$0004
 FDB $04CD,$0003
 FDB $0666,$0002
 FDB $07FF,$0002
 FDB $FFFF,$0002
* END OF TABLE
*
* THIS IS THE VELOCITY STEPPER MOTOR CONTROL
* IT USES B3 FOR THE OVERALL GAIN CONSTANT
VALG JSR PIDCC ;CALL PID CONTROL
 SUB B LCNTL+1 ;SUBTRACT THE LAST CONTROL
 SBC A LCNTL
 STA A LCNTL ;AND SAVE FOR MULTIPLY
 STA B LCNTL+1
 PSH A
 PSH B
 JSR SMOUT ;OUTPUT TO THE D/A ALSO
 PUL B
 PUL A
 JSR FMLT ;TIMES B3
 FDB LCNTL
 FDB B3 ;NUMBER OF STEPS PER VOLT
 TST A
 BPL VALG1 ;IT IS POSITIVE
 COM A
 COM B
 ADD B #1
 ADC A #0 ;NEGATE A,B
```

```
 PSH A
 LDA A OVLD ;MAKE DIRECTION = 0
 AND A #$FE
 STA A OVLD
 PUL A
 BRA VALG2
VALG1 PSH A
 LDA A OVLD ;SET DIRECTION = 1
 ORA A #$01
 STA A OVLD
 PUL A
VALG2 TST A ;IS IT SATURATED?
 BEQ VALG3
 LDA B #$FF
VALG3 STA B STEPS ;B HAS NUMBER OF STEPS FOR MOTOR
 LDA A #TSTC  TICR  TOEN
 STA A TIMER ;START PULSES TO MOTOR
 JSR PIDCD ;CLEAN UP PID
 RTS ;EXIT
*
* THIS IS THE MIDDLE ADDRESS ROM
 ORG $C800
*
* THIS IS A ROUTINE FOR THE QUICK
* MULTIPLICATION OF TWO 16 BIT NUMBERS
* ONE OF WHICH IS IN FLOATING POINT
*
* THE FLOATING POINT NUMBER IS CALLED MPLY
* AND IS IN THE FOLLOWING FORMAT
* BYTE 0 BIT 7 IS THE MANTISSA SIGN
* BITS 6-0 ARE THE TWOS COMP CHARACTERISTIC
* BYTES 1,2 IS THE BINARY MANTISSA
*
* THE BINARY NUMBER IS CALLED MCND
* AND IS IN THE FOLLOWING FORMAT
* BYTE 0 IS THE MSB
* BYTE 1 IS THE LSB
*
* THE RESULT IS RETURNED IN A AND B (MSB,LSB)
*
* THE CARRY BIT WILL BE SET IF OVERFLOW OCCURED
*
* NO REGISTERS ARE SAVED
* THE CALLING SEQUENCE IS;
* JSR FSMT
* FDB MCND
* FDB MPLY (MULTIPLIER AND MULTIPLICAND ADDRESSES)
* RETURN
*
* THE CODE IS REENTRANT, ALL INTERMEDIATE
* VALUES ARE HELD ON THE STACK
*
* DURING EXECUTION THE STACK APPEARS
*
```

```
* STACK POINTER
*  CNT (THE CYCLE COUNTER)
*  MCSIGN
*  MPSIGN
*  CHAR  (MPLY)
*  MSB
*  LSB
*  MSB  (MCND)
*  LSB
*  RETURN MSB
*  RETURN LSB
*
*
MOVE JMP MOVER ;ENTRY FOR MOVE ROUTINE
GHDIN JMP GHD ;ENTRY FOR HEX DIGIT GETTER
KININ JMP KINEN ;ENTRY FOR HEX TO DECIMAL
STRT JMP START ;THE RESET SEQUENCE
**********************************
* GET ARGS AND SET UP STACK
**********************************
MLT TSX
 LDX 0,X ;GET ADDRESS OF FDB
 LDX 0,X ;GET MCND ADDRESS
 LDA A 1,X ;PUT MCND ON STACK
 PSH A
 LDA A 0,X
 PSH A
*
 TSX ;PUT MPLY ON STACK
 LDX 2,X
 LDX 2,X
 LDA A 2,X
 PSH A
 LDA A 1,X
 PSH A
 LDA A 0,X
 PSH A
 CLR A ;CLEAR SOME SPACE ON STACK
 PSH A
 PSH A
 LDA A #16 ;16 CYCLES OF SHIFT
 PSH A
 TSX ;X POINTS TO STACK
*
 ASL 3,X ;GET SIGN OF MPLY
 BCC MBN
 INC 2,X
*
MBN TST 6,X ;MAKE MCND BINARY AND
 BPL ROMM ;SAVE SIGN ON STACK
 INC 1,X
 LDA A 6,X
 LDA B 7,X
 COM A
```

```
 COM B
 ADD B #1
 ADC A #0
 STA A 6,X
 STA B 7,X
*
ROMM LDA A 1,X ;SIGN OF RESULT IN MCSIGN
 EOR A 2,X
 CLR 2,X ;CLEAR FOR OVERFLOW SIGN
 STA A 1,X
*
* DO SHIFT AND ADD MULTIPLY
*
 CLR A
 CLR B
 ROR 4,X
 ROR 5,X
ROM BCC RTM
 ADD B 7,X
 ADC A 6,X
RTM ROR A
 ROR B
 ROR 4,X
 ROR 5,X
 DEC 0,X
 BNE ROM
*
 PSH A ;PUT 10 IN CNT FOR INCREMENTING
 LDA A #10 ;STACK POINTER AT END
 STA A 0,X
 PUL A
*
* GET CHAR IN TWO'S COMP. SIGN IS ALREADY STRIPPED
*
MPL ASR 3,X
 BEQ MNRND ;NO SHIFTS IF EXP 0
 BPL MLSH
MRSH BSR RSH ;SHIFT RESULT RIGHT
 BSR RSH ;ON NEGATIVE EXP
 BSR RSH
 BSR RSH
 INC 3,X
 BNE MRSH
 BRA MNRND
RSH LSR A
 ROR B
 ROR 4,X
 RTS
MLSH BSR LSH ;SHIFT RESULT LEFT ON
 BSR LSH ;POSITIVE EXP
 BSR LSH
 BSR LSH
 DEC 3,X
 BNE MLSH
```

```
 BRA MNRND
LSH ASL 5,X
 ROL 4,X
 ROL B
 ROL A
 BPL LSXT ;SET OVERFLOW
 INC 2,X
LSXT RTS
*
MNRND TST 4,X ;SHALL WE ROUND?
 BPL MSGN ;NO
 ADD B #1
 BCC MSGN
 INC A
MSGN TST 1,X ;CHECK SIGN OF RESULT
 BEQ ISP
 COM A ;COMPLEMENT IF NEGATIVE
 COM B
 ADD B #1
 ADC A #0
ISP INS ;INCREMENT SP AND RETURN
 DEC 0,X
 BNE ISP
 TST 2,X ;SET CARRY IF OVERFLOW OCCURED
 BEQ NXOV
 SEC
 BRA FMXT
NXOV CLC
FMXT LDX 8,X
 JMP 4,X
*
* THIS IS A GENERAL PURPOSE
* BLOCK MOVE ROUTINE
* THE CALLING SEQUENCE IS
* LDX #DESTINATION
* JSR MOVE
* FCB  BYTE COUNTER
* FDB  SOURCE ADDRESS
* RETURN
*
MOVER STX TDST ;SAVE DESTINATION
 TSX
 LDX 0,X ;X POINTS TO BYTE COUNTER
 LDA B 0,X ;B IS BYTE COUNTER
 LDX 1,X ;GET SOURCE
 STX TSRC ;AND SAVE
MVLP LDA A 0,X ;MOVE IT ALL
 INX
 STX TSRC
 LDX TDST
 STA A 0,X
 INX
 STX TDST
 LDX TSRC
```

```
 DEC B
 BNE MVLP
 TSX
 LDX 0,X
 INS ;RESTORE THAT STACK
 INS
 JMP 3,X ;RETURN
*
* GET DISPLAY AND SEND IT TO HEX TRANSLATOR
* SET THE CARRY IF EXPONENT IS GT 3 OR LT -3
GHD LDX #KIND
 LDA A #RDAI ;AUTOINC READ OF DISPLAY
 STA A KDSP+1
 LDA B #6 ;6 DIGITS
GHD1 JSR SVNSE ;GET DIGIT
 STA A 0,X
 INX
 DEC B
 BNE GHD1
 CMP A #3 ;A HAS THE EXPONENT
 BGT GHDE ;AND IT CAN'T BE GT 3
 CMP A #-3 ;OR LESS THAN -3
 BLT GHDE
 JSR KINEN ;KINHF 0-2 HAS RESULT IN HEX
 CLC ;NO CARRY IF NUMBER GOOD
 RTS
GHDE SEC ;SET CARRY IF EXP OUT OF RANGE
 RTS
*
*
* THIS IS THE ROUTINE TO TRANSLATE
* BCD FLOATING INPUT TO HEX FLOATING
* THE ALLOWED RANGE IS
* 0.001 X 10**-3  TO 0.999 X 10**3
*
* THE KEYBOARD ENTRY MUST BE
* ENTERED INTO KIND AS FOLLOWS;
* KIND  SIGN
*       MSB
*       ISB
*       LSB
*       EXP SIGN
*       EXP
*
* CLEAR SOME SCRATCHPAD
* THIS IS ALSO ENTRY POINT
KINEN CLR A
 ORA A KIND+1 ;FIRST SEE IF ZERO
 ORA A KIND+2
 ORA A KIND+3
 BNE KIN7
 STA A KINHF
 STA A KINHF+1
 STA A KINHF+2
```

```
 CLR KIND
 CLR KIND+4
 CLR KIND+5
 RTS
KIN7 LDX #KINI
 LDA A #9
KIN0 CLR 0,X
 INX
 DEC A
 BNE KIN0
*
* FIND LOCATION TO PUT DIGITS
* IN KINI, THE INTEGER TABLE
* AFTER MULTIPLY BY 10 EXP 6
*
KIN LDA A #3 ;ADD 3 TO EXP
 TST KIND+4
 BMI KIN1
 ADD A KIND+5
 BRA KIN1+2
KIN1 SUB A KIND+5
 LDX #KINI+6 ;POINT TO LSB OF TABLE -3
KIN2 TST A
 BEQ KIN3 ;MOVE DIGIT WHEN EXP=0
 DEX
 DEC A
 BRA KIN2
KIN3 LDA B KIND+1 ;MOVE DIGITS
 STA B 0,X ;INTO
 LDA B KIND+2 ;TABLE
 STA B 1,X
 LDA B KIND+3
 STA B 2,X
*
* CONVERT INTEGER DECIMAL
* INTO INTEGER BINARY
* USING ADD AND TIMES
* TEN ALGORITHM
 LDA A #4 ;CLEAR RAM
 LDX #KINB
KIN4 CLR 0,X
 INX
 DEC A
 BNE KIN4
 LDA A #9 ;DO 9 DIGITS
 LDX #KINI ;POINT TO MSB OF INTEGER
KIN5 LDA B 0,X ;ADD NEXT DIGIT
 ADD B KINB+3
 STA B KINB+3
 LDA B KINB+2
 ADC B #0
 STA B KINB+2
 LDA B KINB+1
 ADC B #0
```

```
 STA B KINB+1
 LDA B KINB
 ADC B #0
 STA B KINB
 INX ;GET NEXT DIGIT
 DEC A
 BEQ KIN6 ;DONE?
*
* IF NOT, MULTIPLY BY TEN
*
 STX XTEMP ;SAVE KINB
 LDX KINB ;IN KINBT
 STX KINBT ;TO BE ADDED
 LDX KINB+2
 STX KINBT+2 ;LATER
 LDX XTEMP
* MULTIPLY BY TEN
 LDA B #2 ;TWO SHIFTS=TIMES 4
 BSR KINSL
 LDA B KINB+3
 ADD B KINBT+3
 STA B KINB+3
 LDA B KINB+2
 ADC B KINBT+2
 STA B KINB+2
 LDA B KINB+1
 ADC B KINBT+1
 STA B KINB+1
 LDA B KINB
 ADC B KINBT
 STA B KINB
 LDA B #1 ;ONE MORE SHIFT
 BSR KINSL
 BRA KIN5 ;DONE
KINSL ASL KINB+3
 ROL KINB+2
 ROL KINB+1
 ROL KINB
 DEC B
 BNE KINSL
 RTS
*
* NOW WE MULTIPLY BY HEX 10C7 X 16**-8
*
KIN6 LDA A #8 ;FIRST CLEAR STACK ROOM
 CLR B
CLSP PSH B
 DEC A
 BNE CLSP
 LDA A #48 ;48 CYCLES
 PSH A
 TSX
* DO MULT
* 0,X IS CYCLES
```

```
* 1,X IS MSB OF RESULT
* 2,X IS NEXT, THEY ARE TEMP IN A,B
* ETC TO 8,X WHICH IS LSB
*
 CLR A
 CLR B
KINM LSR KINB
 ROR KINB+1 ;TEST
 ROR KINB+2 ;NEXT
 ROR KINB+3 ;BIT
 BCC KINM1
 ADD B #$C7
 ADC A #$10
KINM1 ROR A
 ROR B
 ROR 3,X
 ROR 4,X
 ROR 5,X
 ROR 6,X
 ROR 7,X
 ROR 8,X
 DEC 0,X
 BNE KINM
 STA A 1,X
 STA B 2,X
* FORMAT THE RESULT
 LDA A #10
KINF INX ;POINT TO MSB OF ANSWER
 DEC A
 DEC A
 TST 0,X ;FIND THE NON-ZERO DIGIT
 BEQ KINF
* NOW SEE WHICH NIBBLE IS NON-ZERO
 LDA B #$F0
 AND B 0,X
 BEQ KINF1 ;NOT THAT ONE
KINF3 LDA B 2,X
 STX XTEMP
 LDX 0,X
 TST B
 BPL KINF2 ;SHALL WE ROUND?
 INX
KINF2 STX KINHF+1
 LDX XTEMP
 BRA KINFE
KINF1 LDA B #4 ;SHIFT 4 PLACES LEFT
KINF4 ASL 2,X
 ROL 1,X
 ROL 0,X
 DEC B
 BNE KINF4
 DEC A
 BRA KINF3
* NOW ADD SIGN AND STORE EXP
```

```
KINFE ASL A
 LDA B KIND
 ASL B
 ROR A
 STA A KINHF
*
* RESTORE STACK POINTER
*
 LDA A #9
RSSP INS
 DEC A
 BNE RSSP
 RTS
*
START LDS #$03FF
 LDX #$03FF
STLP CLR 0,X
 DEX
 BNE STLP
 LDX #$0401 ;START STEPPER TIMER AT 10 MSEC
 STX TSTP
 INC STTMD+2 ;SET DISPLAY DIGIT FOR 10 MSEC
 INC DZN ;START AT ORDER 1 D(Z)
 LDA A #$34 ;DIVIDE DISPLAY CLOCK BY 20
 STA A KDSP+1
 CLR KDSP+1 ;LEFT ENTRY OF 8 DIGITS
 LDX #DA
 LDA B #4 ;PA1 AND PB1 ARE OUTPUTS
 COM 0,X
 STA B 1,X
 COM 2,X
 STA B 3,X
 STA B 9,X ;PA3 ARE INPUTS
 LDA B #$2E ;PULSE CB2
 STA B 11,X ;USE RISING EDGE OF CB1
* INITIALIZE THE TIMER CHIP
* TIMER 3 IS USED TO GENERATE 1 MSEC
* COUNT INTERVALS WHICH ARE USED BY
* TIMERS 2 AND 3 TO GENERATE INTERRUPTS
* IN STEPPER MOTCR CONTROL TIMER 1 IS USED
* INSTEAD OF TIMER 2 TO GENERATE INTERRUPTS
 LDA A #TCR3
 STA A TIMER
 LDA A #TCR2  TICR ;INTERRUPTS FROM TIMER 2
 STA A TIMER+1
 LDA A #TSTC ;CONTROL TIMER 1 FOR STEPPER
 STA A TIMER
 LDX #$01F3
 STX TIMER+6 ;USE TO GENERATE 1 MSEC COUNTS
* INITIALIZE THE OVERLOAD PIA
 LDA A #1 ;PA0 IS OUTPUT
 STA A OVLD
 LDA A #$2C ;CA2 IS PULSES TO STEPPER MOTOR
 STA A OVLD+1
```

```
 COM OVLD+2
 LDA B #4
 STA B OVLD+3
 JSR CLDSE ;CLEAR THE DISPLAY
 CLR SPFN ;GET THE FIRST BUTTON TOO
 CLI
 JMP BTST
*
* THIS IS THE HIGHEST ADDRESS ROM
 ORG $CC00
*
ULDE JMP ULD ;ENTRY POINT FOR ULD
SVNSE JMP SVNS ;SEVEN SEGMENT DISPLAY ENTRY POINT
CLDSE JMP CLDS
*
BTST BSR BTWT
 TST SPFN ;ANY FUNCTIONS
 BNE BTST1 ;LAST TIME ?
 JSR NFND1
BTST1 CLR SPFN
 BRA BTST
*
* BUTTON SERVICE ROUTINE
*
BTWT LDA A KDSP+1 ;TEST FOR
 AND A #$07 ;BUTTONS
 BEQ BTWT
 LDA A #RB ;GET BUTTON
 STA A KDSP+1 ;IN
 LDA B KDSP ;REGISTER B
 AND B #$3F
 CMP B #$00 ;IS IT CLEAR FUNCTION?
 BNE NLK
 JSR CLDS
 SEC ;SET CARRY TO CANCEL FUNCTION
 RTS
NLK LDX #NTBL
 LDA A #11 ;11 TRIES, 0-9, -
NSCH CMP B 0,X ;SCAN TABLE FOR
 BEQ NFND ;NUMBER
 INX
 INX
 INX
 DEC A
 BNE NSCH
* IF BTWT WAS CALLED FROM FUNCTION
* WE CANT ALLOW ANOTHER FUNCTION
* TO BE CALLED
 TST SPFN
 BEQ FSCH1
 JSR FERR5
 SEC ;CANCEL FUNCTION
 RTS
FSCH1 LDX #FTBL ;MIGHT BE ONE OF
```

```
 LDA A #11 ;11 FUNCTION BUTTONS
FSCH CMP B 0,X
 BEQ FFND
 INX
 INX
 INX
 DEC A
 BNE FSCH
 RTS
FFND LDX 1,X ;CALL FUNCTION
 INC SPFN ;DON'T CALL DISPLAY
 JSR 0,X
NFND CLC
 RTS
* DISPLAY NUMBER
NFND1 LDA B 2,X ;GET SEVEN SEGMENT
 LDA A DCTR
 INC A
 CMP A #6
 BLT NFND2
 CLR A
 BRA NFND3
NFND2 CMP A #4
 BEQ NFND3
 CMP B #$40
 BNE NFND4
 CLR B
 DEC A
 BRA NFNDX
* A HAS ADDRESS OF DISPLAY
* B HAS DATA
DSPOUT TST A ;SET DECIMAL
 BNE DCPT ;IN
 ORA B #$80 ;FIRST DIGIT
DCPT ORA A #WD
 STA A KDSP+1
 STA B KDSP
 RTS
NFND4 BSR DSPOUT
 BRA NFNDX
NFND3 CMP B #$40 ;IS IT - SIGN?
 BEQ NFND4
 PSH B
 CLR B
 JSR DSPOUT
 INC A
 PUL B
 BRA NFND4
NFNDX AND A #$07
 STA A DCTR
 RTS
*
* FCB IS 8279 BUTTON CODE
* FDB IS BINARY AND 7 SEGMENT CODE
```

```
NTBL FCB $02
 FDB $003F
 FCB $0A
 FDB $0106
 FCB $09
 FDB $025B
 FCB $08
 FDB $034F
 FCB $12
 FDB $0466
 FCB $11
 FDB $056D
 FCB $10
 FDB $067D
 FCB $1A
 FDB $0707
 FCB $19
 FDB $087F
 FCB $18
 FDB $096F
 FCB $01
 FDB $8040 ;MINUS SIGN
FTBL FCB $0B
 FDB LOAD ;ENTER K
 FCB $03
 FDB DKP ;DISPLAY K
 FCB $0C
 FDB LDST ;LOAD SAMPLE TIME
 FCB $04
 FDB DSST ;DISPLAY SAMPLE TIME
 FCB $15
 FDB CLIC ;CLEAR INTEGRATOR
 FCB $13
 FDB DSCM ;DISPLAY CONTROL MODE
 FCB $1C
 FDB LDZN ;LOAD D(Z) ORDER
 FCB $14
 FDB DSZN ;DISPLAY D(Z) ORDER
 FCB $1B
 FDB LDCM ;LOAD CONTROL MODE
 FCB $0D
 FDB STPT ;LOAD SET POINT
 FCB $05
 FDB DSPT ;DISPLAY SET POINT
* END OF TABLE FOR FUNCTIONS
*
*
* LOAD CONTROL MODE
LDCM JSR BTWT
 BCC *+3
 RTS
 LDA B 1,X ;GET BINARY
 BMI LDCME ;ERROR IF A MINUS SIGN
 BRA LDCM1 ;GO LOAD IT
```

```
LDCME JSR FERR4 ;NOT A VALID CONTROL MODE
 RTS
LDCM1 CMP B #5 ;CONTROLS FROM 0 TO 5
 BHI LDCME
 STA B CNTMD
 BRA DSCM
* LOAD D(Z) ORDER
LDZN JSR BTWT
 BCC *+3
 RTS
 LDA B 1,X
 BMI LDCME ;ERROR IF A MINUS SIGN
 BEQ LDZN2
 CMP B #4
 BLT LDZN1
LDZN2 JSR FERR4
 RTS
LDZN1 STA B DZN
 BRA DSZN
* DISPLAY CONTROL MODE
DSCM BSR CLDS ;CLEAR DISPLAY
 LDA A #WDAI ;PREPARE TO WRITE DISPLAY
 STA A KDSP+1
 LDA B #%01011000 ;WRITE A SMALL C
 STA B KDSP
 LDA B CNTMD ;WRITE CONTROL MODE
 JSR BTSV
 RTS
*
* DISPLAY D(Z) ORDER
DSZN BSR CLDS ;CLEAR DISPLAY
 LDA A #WDAI ;PREPARE TO WRITE DISPLAY
 STA A KDSP+1
 LDA B #%01011100 ;WRITE A SMALL D
 STA B KDSP
 LDA B DZN ;WRITE THE D(Z) ORDER
 JSR BTSV
 RTS
*
* CLEAR THE DISPLAY
CLDS INC SPFN
 LDA A #CD
 STA A KDSP+1 ;CLEAR
CDC LDA A #5 ;DISPLAY
 STA A DCTR
CDCW TST KDSP+1 ;WAIT FOR IT
 BMI CDCW ;TO HAPPEN
 RTS
*
* CLEAR INTEGRATOR INITIAL CONDITIONS
CLIC LDX #0
 STX ISMPL
 RTS
*
```

```
* ERROR NUMBERS
FERR0 CLR B
 BRA FERR
FERR1 LDA B #1
 BRA FERR
FERR2 LDA B #2
 BRA FERR
FERR3 LDA B #3
 BRA FERR
FERR4 LDA B #4
 BRA FERR
FERR5 LDA B #5
 BRA FERR
*
* DISPLAY ERROR
FERR JSR CLDS ;FIRST CLEAR THE JUNK AWAY
 PSH B
 LDA A #WDAI
 STA A KDSP+1
 LDA A #3
 LDX #ERTB
ERLP LDA B 0,X
 STA B KDSP
 INX
 DEC A
 BNE ERLP
 PUL B
 BSR BTSV
 RTS
* THIS IS THE VECTOR TO WRITE
* 'ERR' IN THE DISPLAY
ERTB FCB $79,$50,$50
*
* ROUTINE TO OUTPUT SEVEN
* SEGMENT DISPLAY FROM
* BINARY CONTENTS OF B
* X AND A ARE SAVED
*
BTSV PSH A
 STX PPX
 LDA A #17
 LDX #NTBL
BTSLP CMP B 1,X
 BEQ BTSFN
 INX
 INX
 INX
 DEC A
 BNE BTSLP
BTSFN LDA B 2,X
 PUL A
 LDX PPX
 STA B KDSP
 RTS
```

```
*
* ROUTINE TO DISPLAY K'S
DKP JSR BTWT
 BCC *+3
 RTS
 LDA B 1,X ;GET BINARY
 BMI DKPE
 LDX #COTBLT
 BSR ULD
* ALSO ENTRY FOR OTHER DISPLAY PROGRAMS
DKP1 LDA A #WDAI ;AUTOINC WRITE
 STA A KDSP+1 ;TO DISPLAY
 INC DCML ;ADD DECIMAL
 BSR DKP3 ;AND DISPLAY BLANK OR -
 LDA A #3
 BSR DKP4
 BSR DKP3
 LDA A #1
 BSR DKP4
 RTS
DKP3 CLR B
 TST 0,X
 BPL DKP2
 LDA B #$40
DKP2 TST DCML
 BEQ DKP21
 ORA B #$80 ;ADD DECIMAL
 CLR DCML
DKP21 STA B KDSP
 INX
 RTS
DKP4 LDA B 0,X
 BSR BTSV
 INX
 DEC A
 BNE DKP4
 RTS
DKPE JSR FERR0
 RTS
*
* UTILITY ROUTINE FOR LOAD AND DISPLAY
* ADD 2 TIMES B TO X AND MAKE X
* POINT TO DESIRED K
*
ULD TST B
ULD2 BEQ ULD1
 INX
 INX
 DEC B
 BRA ULD2
ULD1 LDX 0,X
 RTS
*
COTBL FDB KP,KI,KD,A0,A1,A2,A3,B1,B2,B3
```

```
COTBLT FDB KPT,KIT,KDT,A0T,A1T,A2T,A3T,B1T,B2T,B3T
*
* ROUTINE TO LOAD COEFFICIENTS
* INTO STORAGE FROM THE
* DISPLAY RAM
LOAD JSR BTWT
 BCC *+3
 RTS
 LDA B 1,X ;GET COEFF. NO.
 BMI LOADE
 PSH B ;WE NEED IT AGAIN LATER
 PSH B ;TWICE
 JSR GHDIN ;GET HEX FROM DISPLAY
 BCC LOAD1 ;IF CARRY IS SET EXP WAS OUT OF RANGE
 PUL B ;SO WE RESTORE STACK
 PUL B ;AND PRINT ERROR MESSAGE
 JSR FERR1
 RTS ;AND EXIT
LOAD1 PUL B
* MOVE COEFF INTO STORAGE
 LDX #COTBL
 BSR ULD
 SEI
 JSR MOVE
 FCB 3
 FDB KINHF
 CLI
* SAVE DISPLAY
 PUL B
 LDX #COTBLT
 BSR ULD
 JSR MOVE
 FCB 6
 FDB KIND
 JSR CLDS ;CLEAR DISPLAY
 RTS ;AND RETURN
LOADE JSR FERR0
 RTS
*
* ROUTINE TO SCAN TABLE FOR
* 7 SEGMENT DISPLAY AND
* RETURN BINARY NO IN A
* CARRY IS SET IF NO ENTRY FOUND
* 80 IS RETURNED IF - SIGN FOUND
* ROUTINE GETS SEARCH VALUE
* FROM THE DISPLAY RAM, WHICH
* HAS ALREADY BEEN SET UP
* FOR AUTO INCREMENT
* B IS PRESERVED AS WELL AS X
* N IS SET IF - SIGN RETURNED
*
SVNS PSH B
 STX PPX
 LDA B KDSP ;GET DISPLAY
```

```
 AND B #$7F ;MASK OFF DECIMAL
 LDA A #17 ;17 TRIES
 LDX #NTBL
SSRC CMP B 2,X ;SCAN THE TABLE
 BEQ SVFN ;FOUND IT
 INX
 INX
 INX
 DEC A
 BNE SSRC
 SEC
 BRA SVND
SVFN CLC
 LDA A 1,X ;RETURN BINARY
SVND PSH A
 TPA
 LDX PPX
 TAP
 PUL A
 PUL B
 RTS
*
* ROUTINES TO LOAD SAMPLE TIME
* FROM KEYBOARD IN MS
*
* HEX 1000 CONSTANT
STCN FDB $03E8
*
LDSTB JMP LDSTE
LDST JSR BTWT ;FIND WHICH TIME
 BCC *+3
 RTS ;ERROR
 LDA B 1,X ;GET BINARY NUMBER
 PSH B ;SAVE IT
 BMI LDSTB ;NEGATIVE BAD
 CMP B #2 ;IS IT FOR THE STEPPER?
 BEQ STTM
 CMP B #1 ;MUST BE 0 OR 1 NOW
 BHI LDSTE
 JSR GHDIN ;GET HEX FROM DISPLAY
 BCS LDSTE2 ;EXP OUT OF RANGE IS CARRY SET
 ASL KINHF ;MAKE POSITIVE
 LSR KINHF
 CLR KIND
 JSR FMLT ;MULT BY 1000
 FDB STCN ;TO GET MSEC
 FDB KINHF
 BCS LDSTE2
* NOW CHECK IF OUT OF RANGE
 TST A ;IS A,B LT 10 MSEC
 BNE LDST1 ;A IS NOT ZERO
 CMP B #10 ;IS B LT 10
 BCS LDSTE2
 BRA LDST2 ;IT IS GOOD
```

```
LDST1 CMP A #$75 ;IS A,B GT 30 SEC?
 BHI LDSTE2 ;BAD IF GT
 BNE LDST2 ;IT IS GOOD
 CMP B #$60 ;BAD IF GT
 BHI LDSTE2
LDST2 SUB B #1 ;LOAD TIMER
 SBC A #0
 TSX
 TST 0,X ;IS THIS TIMER 0 OR 1
 INS ;RESTORE STACK
 BNE LDST3
 STA A TIMER+4
 STA B TIMER+5
 STA A TMSV
 STA B TMSV+1
 LDX #SMPT ;SAVE DISPLAY
 BRA LDST4
LDST3 STA A TMSV+2
 STA B TMSV+3
 LDX #SMPTT
LDST4 JSR MOVE
 FCB 6
 FDB KIND
 JSR CLDS
 RTS
STTM JSR GHDIN ;GET HEX FROM DISPLAY
 BCS LDSTE2 ;INPUT ERROR
 ASL KINHF ;MAKE POSITIVE
 LSR KINHF
 CLR KIND
 JSR FMLT ;MULTIPLY BY 1000
 FDB STCN ; TO GET MSEC
 FDB KINHF
 BCS LDSTE2 ;OUT OF RANGE
 TST A ;CHECK FOR RANGE
 BNE LDSTE2
 CMP B #5
 BLS LDSTE2 ;LESS THAN 6 MS IS TOO FAST
 DEC B
 LSR B ;DIVIDE BY 2
 STA B TSTP ;SAVE FOR STEP TIMER
 LDX #STTMD ;SAVE DISPLAY
 INS ;RESTORE STACK
 BRA LDST4
LDSTE PUL B
 JSR FERR0
 RTS
LDSTE2 PUL B
 JSR FERR2
 RTS
*
* CONSTANT FOR 12 BIT CONVERSION
TWBC FCB $00
 FDB $1996
```

```
*
* LOAD THE SET PCINT
* THE RANGE IS -10 TO +10 VOLTS
* A VALUE LESS THAN 4.88 MVOLTS IS ERROR
STPT JSR GHDIN ;GET IN HEX FROM THE DISPLAY
 BCS STPTE ;EXP OUT OF RANGE IF CARRY SET
 LDA A KINHF ;GET EXP
 ASL A ;STRIP MANTISSA SIGN
 ASR A
 DEC A
 BGT STPTE ;EXP TOO BIG FOR SET POINT
 CMP A #-2
 BLT STPTE ;EXP TOO SMALL FOR SET POINT
STPT5 INC A
 BGT STPT2
 LDA B #4 ;NOW WE DO THE HEX SHIFTING
STPT1 LSR KINHF+1
 ROR KINHF+2
 DEC B
 BNE STPT1
 BRA STPT5
* GET RID OF SIGN BIT
* SINCE THIS IS UNSIGNED BINARY
* DIVIDE BY TWO
STPT2 LSR KINHF+1
 ROR KINHF+2
 LDA A #$50 ;HEX CANT BE GT 5000
 CMP A KINHF+1
 BCS STPTE
 BHI STPT3
 TST KINHF+2
 BNE STPTE
* MULTIPLY BY 0.1996 TO GET 12 BIT
STPT3 JSR FMLT ;NUMBER IS OK
 FDB KINHF+1 ;RESULT IN A,B
 FDB TWBC
 TST KINHF ;CHECK SIGN
 BPL STPT4
 COM A
 COM B
 ADD B #1
 ADC A #0
STPT4 PSH B
 PSH A
 SUB B SETPNT+1
 SBC A SETPNT ;GET DIFFERENCE
 STA A PPX ;AND SAVE IN PPX
 STA B PPX+1
 PUL A
 PUL B
 STA A SETPNT ;SAVE IT
 STA B SETPNT+1
 LDA A #4 ;SET BUMPLESS FOR 4 TIMES
 STA A BMPFLG
```

```
 LDX #SETPD ;SAVE DISPLAY
 JSR MOVE
 FCB 6
 FDB KIND
 LDA A DZN
 INC A
 STA A DZPFLG ;SET FOR N+2 DEADBEAT
 JSR CLDS
 RTS
STPTE JSR FERR3
 RTS
*
* DISPLAY SAMPLE TIME
*
DSST JSR BTWT
 BCC *+3
 RTS
 LDA B 1,X
 BMI DSSTE
 BEQ DSST1
 CMP B #2 ;IS IT FOR STEPPER?
 BEQ DSST3
 CMP B #1
 BNE DSSTE
 LDX #SMPTT
 BRA DSST2
DSST1 LDX #SMPT
DSST2 JSR DKP1
 RTS
DSSTE JSR FERR0
 RTS
DSST3 LDX #STTMD
 BRA DSST2
*
* DISPLAY SET POINT
*
DSPT LDX #SETPD
 JSR DKP1
 RTS
* THESE ARE THE INTERRUPT AND
* START VECTORS
 ORG $CFF8
 FDB CNTLP,CNTLP,CNTLP,STRT
 END
```